Häufig beantwortete Fragen im Zusammenhang mit MS Word 97 bzw. 2000 und Word-VBA

Inhalt

Index xvii
A xvii
B - C xviii
D xix
E xx
F xxi
G - J xxii
K - M xxiii
N - Q xxiv
R xxv
S xxvi
T - U xxviii
V - Z xxix

Zur 2. Ausgabe 30

Prolog 31

Verwechselbare Begriffe in MS Word 33
Problem 33
Diese Seite 33
Der Begriff *MS Word* 33
Der Begriff *Zeile* bzw. *Zeilenumbruch* 34
Der Begriff *Spalte* 34
Der Begriff *Vorlage* 35
Die Begriffe *Formular*, *Formularfelder* und *Feld* 36
Der Begriff *Textfeld* 36
Der Begriff *Grafik* bzw. *Objekt* 37
Der Begriff *Textmarke* 38
Der Begriff *Schaltfläche*, *Knopf* oder ähnliche Begriffe 38
Der Begriff *Kontrollkästchen* bzw. *CheckBox* 39
Der Begriff *Listenfeld* bzw. *ListBox* 39
Der Begriff *Kombinationsfeld* oder *ComboBox* 39

Soll man grosse und komplexe Dokumente (z.B. eine Diplomarbeit) in MS Word erstellen? 40
Problem 40
Antwort 40

Ein rotes Kreuz in einem Rechteck erscheint an der Stelle einer Grafik 44
Symptome 44
Ursache 44
Abhilfe 44
Vorbeugung 44
Anmerkungen 45
Daten professionell Sichern 45
Links zu diesem Thema 48

Der Fussnotentext wird nicht auf der richtigen Seite gedruckt 49
Symptome 49
Ursache 49
Abhilfe 49
Anmerkung 49
Links zu diesem Thema 49

Es stehen keine oder nur zwei Schriften zur Verfügung 50

Symptome 50
Ursache 50
Abhilfe 50
Anmerkung 51
Links zu diese, Thema 51

Die Seitenränder liegen ausserhalb des bedruckbaren Bereiches! 52
Wieso kann ich nicht bis zum Blattrand drucken? 52
Problem 52
Antwort 52
Erklärungen 52
Lösung 53
Warnung beim Druck ex VBA unterdrücken 54
Links zu diesem Thema 54

Wie kann ich aus einem Word-Dokument ein Büchlein erzeugen? 55
Problem 55
Antwort 55
Alternativen 55
Persönlich Empfehlung 59
Wieso keine Makrolösung mit Word-VBA? 60
Links zu diesem Thema 60

In der Kopf- oder Fusszeile erscheint die unsinnige Angabe Seite 7 von 1 61
Symptome 61
Ursache 61
Abhilfe 61
Anmerkungen 61
Links zu diese, Thema 63

Word2000: In der Kopf- oder Fusszeile erscheinen die Angaben Seite 1 von 1, Seite 2 von 2, Seite 3 von 3, u.s.w. 64
Problem 64
Erklärung 64
Lösung 64
Rundherumläufe 64
Anmerkungen 64
Links zu diesem Thema 64

MS Word startet nur noch ohne Menüleiste, Symbolleisten und Lineal! 65
Problemstellung 65
Erklärung 65
Lösung 65
Anmerkungen 66
Links zu diesem Thema 66

WD2000: Die Grammatikprüfung lässt sich nicht aktivieren. 67
Problem 67
Voranbemerkungen 67
Lösung 67

Wie kann ich in allen Dokumententeile, also auch in Kopf- und Fusszeile sowie in Textfeldern die Felder aktualisieren? 69
Problem 69
Antwort 69
Lösung 69
Anmerkung 69
Tabelle der «*StoryTypes*» in Word 70
VBA-Code 71

Wie kann ich mit MS Word nicht zusammenhängende Textstellen im Dokument gleichzeitig markieren? 72
Frage 72
Antwort 72
Alternativen 72

Rechtschreibprüfung und andere Funktionen stehen in einem teilweise geschützten Dokument nicht zur Verfügung 75
Problem 75
Antwort 75
Lösung 76
VBA-Code 76
Verändern von Dokumenteneigenschaften bei geschützten Formularen 78
Links zu diesem Thema 78
Wo kann ich das Wörterbuch zur neuen deutschen Rechtschreibung beziehen? 79
Problem 79
Lösung 79
Einschränkungen und Mängel 80
Konvertierung bestehender Dokumente von der alten Rechtschreibung nach der neuen 81
Links zu diesem Thema 81
Wo kann ich Wörterbücher für Fremdsprachen beziehen? 82
Problem 82
Vorbemerkungen 82
Welche Komponenten beinhalten die Rechtschreibetools für Office 97 und Office2000? 82
Wie installiere ich die Rechtschreibetools für Fremdsprachen? 83
Fremdsprachen welche die deutsche Ausgabe von Office 97 unterstützt 83
Fremdsprachen welche die deutsche Ausgabe von Office2000 unterstützt? 83
Wo kann ich die nötigen Dateien für die Rechtschreibeprüfung in weiteren Sprachen erwerben? 84
Anmerkungen 84
Links zu diesem Thema 84
Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren? 85
Frage 85
Antwort 85
Sprachen bzw. Sprachfamilien, welche von Windows 98 unterstützt werden 85
Sprachunterstützung unter Windows (98) installieren 85
Zusätzlicher Tastaturtreiber installieren 86
Links zu diesem Thema 86
Das Öffnen und Schliessen von Word-Dokumenten und anderen Office-Dateien, dauert immer länger 87
Symptome 87
Ursache 87
Abhilfe 88
Anmerkungen 88
Links zu diesem Thema 89
Wie lässt sich die Standardschrift für neue Word-Dokumente nachhaltig ändern? 90
Wie lässt sich die automatische Silbentrennung nachhaltig aktivieren? 90
Wie lässt sich die Zoom-Einstellung des Dokumentenfenster nachhaltig ändern? 90
Symptome 90
Standardschrift nachhaltig ändern 90
Silbentrennung nachhaltig einschalten 91
Zoomfaktor nachhaltig ändern 92
Wie kann ich eigene Untermenüs für AutoTexte erstellen? 93
Symptome 93
Lösung 93
Anmerkungen 94
Links zu diesem Thema 94
Positionsrahmen versus Textfelder 95
Problem 95
Antwort 95
Funktionale Gegenüberstellung Positionsrahmen und Textfelder 95
Probleme im Zusammenhang mit Textfeldern 96
Textfeldern nach Positionsrahmen konvertieren 96

Links zu diesem Thema 96

Das erste Wort nach einem Punkt wird immer mit einem Grossbuchstaben geschrieben 97
Symptome 97
Lösung 97
Umgehung 98

Ich möchte die Seitenzahl auf der ersten Seite nicht anzeigen. 99
Ich möchte für die ersten Seiten im Dokument römische Zahlen anzeigen. 99
Problemstellung 99
Lösung 99
Anmerkungen 101
Links zu diesem Thema 101

Wie drucke ich in meinem Dokument Seitenzahlen nach der DIN? 102
Problemstellung 102
Lösung 102
So bringen Sie einen Feldcode in Ihr Dokument ein 102
Anmerkung 102
VBA-Code 103

Wo werden Bausteine wie Formatvorlagen, AutoTexte, Symbolleisten, AutoKorrektur- und Benutzerwörterbucheinträge gespeichert? 105
Diese Seite 105
Formatvorlagen, AutoTexte und Symbolleisten 105
AutoKorrektur-Einträge 106
Benutzerwörterbücher 109
Links zu diesem Thema 109

Wo speichert MS Word seine Einstellungen (Umgebungsoptionen)? 110
Diese Seite 110
Frage 110
Voranbemerkungen 110
Formatvorlagen als Trägerinnen von Einstellungen 111
Die Bedeutung der Standardvorlage mit dem Namen «*Normal.dot*» 112
Anwendungsweite Einstellungen in der Registrierdatenbank 113
Word-Bausteine 115
Benutzerdefinierte Etiketten 115
Fazit 116
Installationswerte wieder herstellen (Alias: Die zehn Gebote, wenn Word spinnt) 117
Links zu diesem Thema 118

Wie kann ich meine Formatvorlagen übersichtlich organisieren? 119
Problem 119
Antwort 119
Lösungsvarianten 120
Anmerkung 121

Wie referenziere ich das jeweilige Kapitel in der Kopf- oder Fusszeile? 122
Aufgabenstellung 122
Lösung 122

Word findet bei der Generierung des Abbildungsverzeichnisses keine Einträge 123
Symptome 123
Ursache 123
Abhilfe 123
Vorbeugung 124
Anmerkungen 124
Links zu diesem Thema 124

Wie kann ich ein Inhaltsverzeichnis in ein anderes Word-Dokument übernehmen 125
Problem 125
Ursache 125
Lösung 125

VBA-Code 126

Wie kann ich mit MS Word Lückentext erstellen und handlich zwischen Aufgabenstellung und Lösung umschalten? 127
Frage 127
Word-integrierte Lösung 127
Komfortlösung mit VBA 127

Der Dialog Einfügen - Symbol... zeigt statt den Symbolen nur leere Quadrate an 132
Symptome 132
Ursache 132
Vorabklärungen 132
Abhilfe 133
Links zu diesem Thema 133

Word vergisst die Skalierung von verknüpften Grafiken 134
Symptome 134
Abhilfe 134
Anmerkungen 134
Links zu diesem Thema 134

Wie kann ich obsolete Einträge zu Dateien im Arbeitsmenü oder in der Auflistung der zuletzt geöffneten Daten löschen? 135
Symptome 135
Antwort 135
Einrichten des Arbeitsmenüs 135
Zahl der Einträge, welcher unter der Auflistung der zuletzt geöffneten Dateien erscheint, beeinflussen 136
Synchronisation automatisiert erzwingen 136

Es stehen keine Seitenrahmen (Zierrahmen) zur Verfügung! 137
Problem 137
Ursache 137
Lösung 137
Anmerkung 137
Links zu diesem Thema 137

Das Betätigen der Enter-Taste in einem konventionellen Formulartextfeld erzeugt einen Zeilenumbruch und verschiebt die Darstellung in meinem Formular 138
Symptome 138
Erklärung 138
Abhilfe 138
VBA-Code 139
Anmerkung 142
Links zu diesem Thema 142

Wie kann ich leere Absätze (Zeilen) aus einem Dokument entfernen? 143
Wie kann ich Text, bei dem jede Zeile mit einer Absatzmarke oder einer manuellen Zeilenumschaltung endet, in Fliesstext umwandeln? 143
Problem 143
Diese Seite 143
Einleitung 143
Terminologie 143
Lösungen 145
Wie kann ich auch nach anderen Formatierungszeichen suchen lassen? 148
Gegenüberstellung Word 97 - Word2000 bezüglich der Formatierungszeichen 149
Links zu diesem Thema 149

Gegenüberstellung deutschsprachiger Feldnamen in Word 97 und englischsprachiger Feldnamen in Word2000 150
Problem 150
Diese Seite 150
Einleitung 150
Tabelle der Feldnamen in Word 97 bzw. in Word2000 151
Gegenüberstellung der Formatierungszeichen (nicht druckbare Zeichen) Word 97/Word2000 154

Probleme und Ungereimtheiten 154
Felder mit VBA version- und sprachgrenzenüberschreitend einfügen 155
Links zu diesem Thema 157

Arbeiten mit dem Range-Objekt in MS Word 158
Frage 158
Diese Seite 158
Was ist ein Range? 158
Wieso ist die Hemmschwelle beim Arbeiten mit Ranges so gross? 160
Vorteile von Range-Objekten (oder das flexibelste Word-Objekt überhaupt) 161
Verzeichnis der hier präsentierten VBA-Lösungen bzw. Lösungsansätze 163
Tabelle der Word-Objekte, welche über eine Range-Eigenschaft verfügen 164
Ranges ansprechen 166
Tabelle der Dokumententeile (StoryTypes) in MS Word 200
Range-Objekte bilden 201
Arbeiten mit der Eigenschaft FormattedText 204
Arbeiten mit Textmarken 205
Arbeiten mit Zeilen und Seiten 206
Links zu diesem Thema 207

Vom Umgang mit Zeilen und Seiten in MS Word 208
Problem 208
Diese Seite 208
Einleitung 208
Zeilen- und Seitenumbrüche 209
Vorläufiges Fazit 210
Terminologie 210
Das Kreuz mit Kopf-/und Fusszeilen 212
Dank vordefinierten Textmarken Bereiche (*Ranges*) bilden 212
VBA-technischer Umgang mit Zeilen 213
VBA-technischer Umgang mit Absätzen 216
VBA-Beispiel mit Fliesstext 218
Warnung: MS Word steht mit Seiten auf dem Kriegsfuss! 219
VBA-technischer Umgang mit Seiten 221
Tabelle der vordefinierten Textmarken in MS Word 226
Links zu diesem Thema 227

VBA-technischer Umgang mit Word-Tabellen 228
Problem 228
Diese Seite 228
Einleitung 228
Das Objekt-Modell von Word-Tabellen 229
Feststellen, ob die verlangte Tabelle im Dokument vorhanden ist 229
Text aus einer Tabellenzelle auslesen 230
Word-Tabelle in einem Datenfeld (*Array*) abbilden 231
Feststellen in welcher Zelle sich die Einfügemarke befindet 235
Eine bestimmte Tabelle über eine Textmarke auffinden 237
Eine Tabelle mit VBA erstellen 238
Zusätzliche Zeilen oder Spalten in eine bestehende Tabelle einbringen 242
Arbeiten mit Tabellenspalten 244
Die *Range*-Eigenschaft einer Tabelle ansprechen 244
Zellbereich markieren 245
Werte in Zellen formatieren 246
Letzte Tabellenzeile jeder Seite auffinden 246
Formel in eine Zelle der letzen Tabellenzeile einfügen 247
Verschachtelte Tabellen in Word2000 verarbeiten 248
Word-Tabelle auf einer UserForm abbilden 249
Weitere VBA-Splitter im Zusammenhang mit Tabellen 252
Ein Excel-Tabellenblatt in einer Word-Tabelle abbilden 252
Eine Access-Tabelle in einer Word-Tabelle abbilden 254
Links zu diesem Thema 256

VBA-technischer Umgang mit Textmarken 257
Diese Seite 257
Einleitung 257
Arten von Textmarken und ihr Verhalten 257
Anspringen von Textmarken versus Arbeiten mit der *Range*-Eigenschaft von Textmarken 259
Feststellen, ob eine bestimmte Textmarke im Dokument vorhanden ist 260
Textstelle zwischen zwei Textmarken finden bzw. markieren 261
Wie Textmarken die Arbeit erleichtern können 262
Textmarkentext mit benutzerspezifischen Werte füllen 262
Tool für das schnelle Navigieren zwischen Textmarken im Word-Dokument 263
Links zu diesem Thema 263

Wie kann ich in einem VBA-Programm (Makro) feststellen, auf welcher Seite sich die Einfügemarke befindet? 264

Wie kann ich in einem VBA-Programm (Makro) feststellen, ob sich die Einfügemarke in einer Tabelle befindet? 264
Frage 264
Diese Seite 264
Die *Information*-Eigenschaft 264
Feststellen, wie viele Seiten das Dokument umfasst 266
Feststellen, auf welcher Seite sich die Einfügemarke befindet 266
Feststellen, auf welcher Seite sich ein bestimmtest Objekt befindet 268
Feststellen, wie viel Platz auf der Seite noch frei ist 268
Text verkleinern/vergrössern, bis er genau auf eine Zeile bzw. genau auf eine Seite passt 270
Feststellen, ob sich die Einfügemarke in einer Tabelle befindet 271
Feststellen, in welcher Tabellen-Zelle sich die Einfügemarke befindet 271
Feststellen, ob sich die Einfügemarke in einem Textfeld befindet 274

Textgebundene und Textungebundene Grafiken und Objekte («InlineShapes» und «Shapes») 275
Diese Seite 275
Einleitung 275
Woran erkenne ich, ob ich eine «*Shape*» oder eine «*InlineShape*» *vor mir habe?* 276
Wieso macht MS Word die Unterscheidung zwischen «*Shapes*» und «*InlineShapes*»? 277
Wie kann ich beeinflussen, ob eine «*Shape*» oder eine «*InlineShape*» entsteht? 278
Wie kann ich Word dazu bewegen den Standardwert, ob beim Einfügen eine «Shape» oder eine «InlineShape» entsteht ändern? 279
Arten von «Shapes» und «InlineShapes» 279
«Shape» in «InlineShape» umwandeln und v.v. 280
Charakteristika von «Shapes» und «InlineShapes» 281
Beschriftungen und Abbildungsverzeichnis 282
Eine «Shape» oder eine «InlineShape» verschieben 283
Objekte an einer bestimmten Stelle auf der Seite festnageln 284
Objekte, welche sich auf jeder Seite wiederholen in das Dokument einbringen 285
Ein Wasserzeichen in das Dokument einbringen 287
VBA-technischer Umgang mit «Shapes» und «InlineShapes» 288
Tabelle der *«StoryRanges»* (Dokumententeile) in MS Word 295
Links zu diesem Thema 296

Wie kann ich auf die Adressen im Outlook Kontakte-Ordner zugreifen? 297
Ausgangslage 297
Adressbuchfunktion installieren 297
Adressbuchfunktion einrichten 297
Adressbuchfunktion benutzen 297
Anmerkungen 298
Links zu diesem Thema 298

Das Outlook-Feld «Kategorien» steht in der Word-Seriendruckfunktion nicht zur Verfügung. 299
Problemstellung 299
Antwort 299
Lösungsvarianten 299
Links zu diesem Thema 305

Wie setze ich in einem Serienbrief eine bedingte Anrede? 306
Wie kann ich beim Seriendruck - Adressetiketten ein Bedingungsfeld einbringen? 306
Wie kann ich Leerschläge, welche aus leeren Felder resultieren, eliminieren? 306
Frage 306
Antworten 306
Vorarbeiten 307
Bedingungsfeld für zwei Bedingungen einrichten 308
Bedingungsfeld für mehr als zwei Bedingungen einrichten 309
Aus Leerfeldern resultierende Leerschläge vermeiden 310
Wie gibt man eine Feldfunktion in MS Word ein? 311
Der Text in konventionellen Formulartextfeldern wird abgeschnitten oder verschluckt 312
Symptome 312
Erklärung 312
Lösung 312
VBA-Code 312
Links zu diesem Thema 313
Der Befehl Suchen/Ersetzen funktioniert nicht in Kopf- und Fusszeilen und auch nicht in Textfeldern 314
Wie kann ich die Suchen/Ersetzen-Funktion auf ein ganzes Verzeichnis mit Word-Dokumenten wirken lassen? 314
Problem 314
Erklärung 314
Lösung 314
VBA-Code 315
Anmerkungen 318
Die elf Gut-Nachtgeschichten von Word 319
Wie kann ich eine Tabelle ohne Gitter erzeugen bzw. eine solche zum Standard küren? 320
Problem 320
Erklärung 320
Abhilfe 320
Wieso kann ich nicht auf das Adressbuch von Outlook Express 4 oder 5 zugreifen. 322
Symptome 322
Ursache 322
Rundherumläufe 322
Wichtig 323
Links zu diesem Thema 323
Wie kann ich die Adressfelder und deren Reihenfolge, welche ich über den Befehl Einfügen Adresse in mein Dokument einfüge beeinflussen? 324
Aufgabenstellung 324
Lösung 324
Tabelle der Adressfeldnamen 325
Beispiel für eine Adressdefinition 326
AutoText-Eintrag aufnehmen 326
Werkzeug für Word2000 Benutzer zur Festlegung der Adressdefinition 327
Weiterreichende Bedürfnisse (Office Automation) 328
Links zu diesem Thema 328
Wie kann ich alle Kopf- und Fusszeilen eines Dokumentes löschen 329
Problemstellung 329
Vorbemerkung 329
VBA-Code 329
Wie kann ich Kopf- und Fusszeilen um 90 Grad drehen? 331
Problemstellung 331
Vorbemerkung 331
Lösung 331
Links zu diesem Thema 332
Wie kann ich ein Word-Dokument spiegelverkehrt ausrucken? 333
Wie kann ich Text in einem Word-Dokument auf den Kopf stellen? 333

Problem 333
Antwort zum spiegelverkehrtem Druck 333
Text auf den Kopf stellen 334
VBA-Routine 335

Word bietet mir nur 16 Schriftfarben an. Kann ich das ändern? 338
Problem 338
Antwort 338
Anmerkungen 338
VBA 338

Wie bringe ich eine Faltmarke in ein Word-Dokument ein? 339
Problemstellung 339
Lösung 339
Anmerkungen 339
VBA-Code 340

Wie kann ich kann ich den Dateiname eines Dokumentes mit Pfadangabe in die Kopf- oder Fusszeile setzen? 342
Problemstellung 342
Lösung 342

Wie kann ich ein Symbol kreieren (z.B. ein x mit Überstrich, wie es in statistischen Berechnungen gebraucht wird)? 344
Problemstellung 344
Lösung 344
Anmerkungen 345

Wie kann ich Nullwerte beim Berechnung unterdrücken? 346
Problemstellung 346
Lösung 346
Anmerkungen 346

Wie kann ich auf fünf (Rappen) genau runden? 347
Problemstellung 347
Lösung 347
Anmerkungen 347

Datums- und Stundenberechnung mit Feldern, Textmarken, Formularfeldern und Tabellen 348
Problemstellung 348
Diese Seite 348
Vorbemerkungen 348
Einfache Datumsberechnung mittels dynamischem Feldcode 349
Komplexer Feldcode zur Datumsberechnung von *Chris Woodman* 349
Allgemeine Felder (Platzhalter) 350
Kalenderwoche Feld-Funktion selbst gemacht 353
Mit Textmarken rechnen 357
Mit konventionellen Formulartextfeldern rechnen 359
Datumsberechnungen in einer Word-Tabelle 361
Links zu diesem Thema 363

Wie kann ich eine Fussnotenzahl doppelt vergeben? 364
Problemstellung 364
Lösung 364
Anmerkungen 364
Links zu diesem Thema 364

Wie kann ich die Fussnotentrennlinie formatieren bzw. löschen? 365
Problem 365
Antwort 365
Links zu diesem Thema 365

Wie kann ich Symbole suchen und ersetzen? 366
Problemstellung 366
Lösung 366

Links zu diesem Thema 366

Wie kann ich Etiketten mit einer Grafik (Logo) versehen? 367
Problem 367
Lösung 367
Anmerkungen 368
Links zu diesem Thema 369

Wie kann ich in MS Word eine Datei unter einem fremdem Format öffnen oder abspeichern? 370
Die Umlaute in einem DOS-Text werden beim Öffnen unter Word falsch dargestellt! 370
Wie kann ich eine Vielzahl von Dokumenten in ein anders Format bringen? 370
Zusätzliche Text- und Grafikfilter von Microsoft 370
Text- und Grafikformate, für welche Microsoft keine Filter zur Verfügung stellt 370
Aus Word-Dokumenten PDF-Dateien erstellen 370
PDF-Dateien oder Teile davon ins Word-Format übernehmen 370
Dateiformate auf Grund ihrer Dateierweiterung identifizieren 370
VBA-technischer Umgang mit internen und externen Textkonverter 370
Diese Seite 370
Was ist ein Konverter bzw. ein Filter? 370
Die Methoden zur Darstellung von formatiertem Text 371
Potential und Grenzen beim Konvertieren von formatiertem Text 372
Datei explizit unter einem bestimmten Format Öffnen 373
Datei explizit unter einem bestimmten Format speichern 374
MS-DOS-Datei mit MS Word öffnen 375
Batch-Konvertierung von Dateien (Stapelverarbeitung) 376
Zusätzliche, aktuelle Text- und Grafikfilter für Office 97 und Office2000 377
Verbreitete Text- und Grafikformate für welche Microsoft keine Filter bzw. Konverter bereit stellt 380
Aus MS Word-Dokumenten PDF-Dateien erzeugen 381
PDF-Dateien oder Teile davon ins Word-Format übernehmen 383
Dateiformat auf Grund ihrer Dateierweiterung identifizieren 384
VBA-technischer Umgang mit externen Word-Konvertern 388
Links zu diesem Thema 391

Ist der Passwortschutz für Word 97 und Word2000 Dokumente sicher? 392
Frage 392
Antwort 392
Anmerkungen 392
Gegebenheiten in Frankreich 392
Passwortknacker 393
VBA-Code zur Generierung eines Zufallcodes 393
Anmerkungen für Besserwisser 394

Wie kann ich einem Word-Dokument oder in einem Word-Dokument eine fortlaufende Nummer vergeben? 395
Problemstellung 395
Diese Seite 395
Lösungsweg 395
Das Dokument x-Mal zu drucken unter Vergabe einer sichtbaren Laufnummer im Dokument 396
VBA-Code zur Vergabe einer Laufnummer bei der Generierung eines Dokuments, welches auf eine bestimmte Dokumentvorlage beruht. 399
Links zu diesem Thema 403

Wie kann ich das Kreuz zum Schliessen einer UserForm in der Titelleiste unterdrücken? 404
Problem 404
Antwort 404
Lösungen 404

Benutzerspezifische Standardwerte für konventionelle Formularfelder, Textmarkentext und UserForm-Controls 408
Diese Seite 408
Aufgabenstellungen 408
Lösungsvarianten 408

Konventionelle Textformularfelder mit benutzerspezifischen Standardwerten belegen 409
Textmarkentext mit benutzerspezifischen Standardwerten belegen 414
Textfelder, Optionsfelder und Kontrollkästchen auf einer *UserForm* mit benutzerspezifischen Standardwerten belegen 422
Aus einer Excel-Tabelle eine INI-Datei generieren 429
Links zu diesem Thema 431

Werte von Textfeldern, Kontrollkästchen und Optionsfeldern einer UserForm speichern und beim nächsten Aufruf der UserForm einsetzen? 432
Aufgabenstellung 432
Lösungsvarianten 432
Charakteristiken (Vor- und Nachteile) der hier präsentierten Lösungen 432
Empfehlung 433
Allgemeine Mimik für alle drei Methoden 433
Allgemeine Mimik für alle drei Methoden 433
Vorarbeiten für alle drei Methoden 434
In Zusammenarbeit mit der Registrierdatenbank 435
In Zusammenarbeit mit einer INI-Datei 436
In Zusammenarbeit mit Dokumentenvariablen 439
Links zu diesem Thema 441

Wie kann ich Schaltflächen einer UserForm dynamisch, während der Laufzeit erstellen? 442
Diese Seite 442
Ausgangslage 442
Antwort 442
Mögliche Einsatzgebiete und Vorteile von dynamisch erstellten UserForms 443
Dynamische Schaltflächen ohne Ereignisroutinen erstellen 443
Tabelle der ProgID's (Klassennamen der Schaltflächen) 444
Dynamische Schaltflächen mit Ereignisroutinen erstellen 444
Dynamische einen Rahmen, welcher Schaltflächen beinhaltet erstellen 447
Textmarkentext über eine UserForm eingeben 448
Tabelle auf einer UserForm abbilden 452
UserForm Schaltflächen für ein Flugbuchungsformular dynamisch generieren 455
UserForm gänzlich dynamisch generieren 459
VBA und Generische Ereignisprozeduren 461
Grenzen einer UserForm 464

Wie kann ich verhindern, dass Word2000 für jedes Dokument welches ich öffne, die Anwendung ein weiteres Mal startet? 465
Symptome 465
Ursache 465
Lösung 465
Anmerkungen 465
Das Schliessen-Kreuz 466
VBA-Code 466
Links zu diesem Thema 466

Wie kann ich einzelne Abschnitte oder einzelne Seiten in Word2000 drucken? 467
Wie kann ich einzelne Abschnitte aus einem Seriendruckdokument drucken? 467
Problemstellung 467
Lösung 467
Serienbrief 467
Anmerkungen 467

Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln? 468
Wie kann ich eine Dezimalzahl in eine hexadezimale Zahl umwandeln? 468
Wie kann ich einen (Geld)-Betrag in Worte umwandeln? 468
Aufgabenstellungen 468
Diese Seite 468
Worum geht es? 468
Wie gibt man eine Feldfunktion in MS Word ein? 469
Feldfunktion um eine Arabisch Zahl in eine Römische Zahl umzuwandeln 469

Feldfunktion um eine Dezimalzahl in eine hexadezimale Zahl umzuwandeln 470
Feldfunktion um eine Zahl (Geldbetrag) in Worten umzuwandeln 470
VBA-Code zur Generierung von römischen Zahlen aus arabischen Zahlen 471
VBA-Code zur Generierung von hexadezimalen Zahlen aus dezimalen Zahlen 474
VBA-Code um eine Zahl (Geldbetrag) in Worte umzuwandeln 475
Formatieren von arabischen Zahlen (Allgemeine numerische Formatschalter für Felder) 483
Nullwerte bei Berechnungen unterdrücken 484
Wie gibt man eine Feldfunktion in MS Word ein? 484
Links zu diesem Thema 484

Wie kann ich ein Beispieltext für alle vorhandenen Schriften anzeigen oder ausdrucken? 485
Aufgabenstellung 485
Lösung 485

Wie kann ich eruieren, welche Schriften in einem Dokument zur Anwendung kommen? 487
Aufgabenstellung 487
Lösung 487

Wie kann ich ein Kapitel aus einem gegliederten Dokument drucken? 490
Problem 490
Antwort 490
Lösung 491

Wie lassen sich Textmarken in einem Word-Dokument bequem ansteuern? 498
Frage 498
Antwort 498
Funktioneller Vergleich der Lösungen 498
Das Textmarken-Utility 499
Links zu diesem Thema 502

MS Word und Schriften 503
Problem 503
Diese Seite 503
Ohne grafikfähigen Druckertreiber läuft gar nichts! 503
Zusätzliche Schriften ordnungsgemäss installieren 504
Wie viele Schriften darf ich maximal unter Windows installieren? 504
Schriften im Word-Dokument einbetten 505
Schriften, welche von MS Word nicht mehr unterstützt werden 506
Zusätzliche Werkzeuge von Microsoft für den Schriftenordner 507
Proportionale versus proportionale Schriften 508
Symbolschriften 509
Zusätzliche Schriften von Microsoft, welche das Euro-Symbol unterstützen 510
Links zu diesem Thema 510

Wissenswertes über die Zeichensatztabellensuppe 511
Diese Seite 511
Von Bit und Byte 511
Der alte ASCII 7-Bit Code 511
Der erweiterte ASCII-Code, EBCDIC und ANSI (8-Bit-Code) 512
Standard- und Symbolschriften 513
Die Schichten («*Layers*») im Zusammenhang mit Zeichentabellen 514
Unicode oder «*Universal Character Set*» (UCS) 515
Links zu diesem Thema 517

Wie lautet der ANSI-Wert zu den in Word gebräuchlichen Sonderzeichen, wie geschütztes Leerzeichen geschützter Trennstrich, etc. 518
Wie kann ich den ANSI (ASCII) Code eines (Sonder-)Zeichens in einem Word-Dokument ermitteln? 518
Wie kann eine Tabelle sämtlicher ANSI (ASCII) Codes umgesetzt in einer bestimmten Schriftart oder Symbolschrift erstellen? 518
Aufgabenstellung 518
Diese Seite 518
Einleitung 518
ANSI-Code für die meist gebräuchlichen Word-internen Sonderzeichen 519

ANSI-Wert einer markierten Zeichenfolge anzeigen 520
Tabelle mit allen potentiell zur Verfügung stehenden Zeichen inklusive dem dazugehörigen ANSI-Wert generieren 524
ANSI-Zeichen über die Tastatur eingeben 527
Umrechnen einer hexadezimalen Zahl in das Dezimalsystem 528
Der Tastaturtreiber und seine Rolle 529
Links zu diesem Thema 529

Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen? 530
Aufgabenstellung 530
Einleitung 530
Unicode-Unterstützung unter Office2000 installieren 530
Schriften ordnungsgemäss installieren 531
Unicode-Zeichen standardmässig über Word-Dialog eingeben 531
VBA-Routine zur Darstellung der Unicode-Familien 531
Problembehandlung im Zusammenhang mit der Darstellung der Unicode-Zeichen 545
Links zu diesem Thema 545

Office2000: Wie kann ich die Word-Dialoge *Datei - öffnen*, *Datei - speichern unter*, *Grafik - einfügen*, etc. anpassen? 546
Frage 546
Antwort 546
Resultat der Anpassung 547
Anmerkungen 547
Links zu diesem Thema 547

Der Word-Dialog «*Datei - Neu...*» soll statt grossen Symbolen kleine Symbole anzeigen 548
Problem 548
Antwort 548
Lösung 548
Anmerkungen 549

Wie kann ich einen Dialog anzeigen, welcher es dem Benutzer erlaubt, einen Ordner bzw. eine Datei auszuwählen? 550
Frage 550
Diese Seite 550
Antwort 550
Dialog zur Auswahl eines Ordners mittels Word-Dialog 551
Dialog zur Auswahl einer Datei mittels Word-Dialog 552
Dialog zur Auswahl eines Ordners oder einer Datei (Windows-API) aus einem Standardmodul 553
Dialog zur Auswahl eines Ordners oder einer Datei (Windows-API) aus einer UserForm 555
Links zu diesem Thema 557

Wie kann ich feststellen, wie oft ein bestimmtest Wort im Dokument vorkommt? 558
Wie kann ich beim wiederholten Ersetzen einer Zeichenfolge ausweisen, wie viele Ersetzungen getätigt wurden? 558
Wie kann ich eine Worthäufigkeitsstatistik eines Word-Dokumentes erstellen? 558
Diese Seite 558
Ausweisen, wie oft eine vorgegebene Zeichenfolge im Hauptteil des Dokumentes auftaucht 558
Ausweisen, wie viele Ersetzungen getätigt wurden 561
Worthäufigkeitsstatistik eines Word-Dokumentes erstellen 563
Links zu diesem Thema 567

Wie kann ich die Verzeichnisstruktur einer Festplattenpartition, CD-ROM, Diskette oder eines Hauptverzeichnisses in einem Word-Dokument ausgeben? 568
Problem 568
Lösung 568
Funktionalität 568
Lösung 569
Links zu diesem Thema 577

Symbolleistenschaltflächen und Menüleisten selbst gemacht. 578
Diese Seite 578
Ausgangslage 578

Arten von Schaltflächen 579
Einfacher Schaltknopf 581
Kombinationsfelder (ComboBoxes) 588
Menü-Leiste 594
Die FindControl-Methode 602
Die ActionControl-Methode 603
Der CustomizationContext 605
Undokumentierte Typen von Schaltflächen 605
Word-Interne Bezeichnung der Symbolleisten 607
Symbolleisten schützen 609
Links zu diesem Thema 611

Wie kann ich alle VBA-Module (Makros) einer Vorlage oder eines Dokuments drucken? 612
Problemstellung 612
Lösung 612

Welche Schalter (Switches) gibt es um die Word-Anwendung zu starten? 615
Frage 615
Antwort 615
Anmerkung 616
Links zu diesem Thema 616

Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen? 617
Wie kann ich aus anderen Anwendungen auf MS Word zugreifen? 617
Diese Seite 617
Vorausbemerkungen 617
Excel mit einer leeren Arbeitsmappe starten 618
Excel mit einer bestimmten Vorlage (XLT) starten 618
Bestehende Excel-Arbeitsmappe öffnen 619
Ganzes Excel-Tabellenblatt in ein *Array* (Datenfeld) laden 619
MS Word aus Excel mit einem leeren Dokument starten 620
Bestehendes Word-Dokument aus Excel öffnen 621
Verweise auf Objektbibliotheken setzen 623
Automatisieren - aber richtig! 624
Von MS Word auf Access zugreifen 625
Mit *«DAO»* Daten aus einem Excel-Tabellenblatt ziehen, ohne die Arbeitsmappe zu öffnen 631
Von MS Word auf MS Outlook2000 zugreifen 633
MS Word Seriendruck aus dem Internet Explorer automatisiert ausführen 638
Links zu diesem Thema 640

Wie lese oder setze ich mit VBA den Wert einer Dokumenteneigenschaft? 641
Problem 641
Diese Seite 641
Mittels VBA auf Dokumenteigenschaften zugreifen 641
Tabelle der word-integrierten Dokumenteneigenschaften 642
Benutzerdefinierte Dokumenteneigenschaften 643
Dokumenteneigenschaften *offline* lesen 644
Links zu diesem Thema 658

Vom Umgang mit Dokumentvariablen in MS Word (einschliesslich Tool zur Verwaltung von Dokumentvariablen) 659
Frage 659
Diese Seite 659
Antwort 659
Ziel und Zweck von Dokumentvariablen 659
VBA-technischer Umgang mit Dokumentvariablen 660
Den Wert einer Dokumentvariable im Dokument anzeigen 663
Den Wert einer Dokumentvariable im Dokument abfragen und darauf reagieren 663
Tool zur Verwaltung von Dokumentvariablen 664
Links zu diesem Thema 672

Wie kann ich bei zeitlich langandauernden VBA-Prozessen eine Fortschrittsleiste darstellen? 673
Problemstellung 673

Vorausbemerkungen 673
Lösungsvarianten 674

Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen? 680
Wie kann ich eine Liste der Word-Befehle erstellen? 680
Frage 680
Antwort 680
VBA-Code 681
Links zu diesem Thema 690
Anmerkung zu den Links 690

Wie kann ich den Office Assistenten programmieren? 691
Ausgangslage 691
Diese Seite 691
Antwort 691
Ausstattungsmerkmale des Office-Assistenten 692
Die wichtigsten Parameters (Eigenschaften) um den Office Assistenten zu steuern 705
Links zu diesem Thema 708

Wo kann ich den Office2000 Service-Release 1 beziehen? 709
Frage 709
Antwort 709
Verbesserungen 710
Bekannte Probleme 711
Links zu diesem Thema 711

Hilfe! - Mein Word schmiert ab! (Alias: Die dreizehn Gebote, wenn Word spinnt! 712
Diese Seite 712
Frage 712
Vorgehen 712
Liste von bekannten und verbreiteten MS Word Abstürze 712
Schema um unspezifische Abstürze der Word-Anwendung zu begegnen 715
Anmerkungen zu Abschmierereien von MS Word 717
Rückschlüsse aus dem Namen des Moduls auf die fehlerhafte Komponente ziehen 719
Wie halte ich die Fehlermeldung fest 720
Unbrauchbare Fehlermeldungen 721
Rechner im abgesicherten Windows-Modus starten 722
Feststellen, ob ein Konflikt mit einem Hintergrundprogramm besteht 724
Bausteine aus der defekten «*Normal.dot*» zurück gewinnen 725
«*Word-Data*» Schlüssel in der Registrierdatenbank umbenennen bzw. löschen 726
Der Umgang mit defekten Dokumenten 727
Office2000 reparieren 731
Office 97 neu installieren 731
«Word-Options» Schlüssel in der Registrierdatenbank umbenennen 732
AutoKorrektur-Dateien und Benutzerwörterbücher umbenennen 732
Hilfe von der Microsoft Online-Datenbank 733
Empfehlenswerte Foren und News-Gruppen 734
PC «Hot Lines» 735
Probleme mit ausschneiden, kopieren, einfügen 736
Absturz beim Drucken und Druckverwandten Funktionen 737
Links zu diesem Thema 737

Wie kann Office 97 oder Office2000 möglichst vollständig von meinem Rechner entfernen? 738
Problem 738
Problem 738
Antwort 738
Links zu diesem Thema 739

So bringen Sie ein Makro ein und verknüpfen es mit einer Schaltknopf auf einer Symbolleiste Ihrer Wahl 740
Frage 740
Antwort 740
Makro und Schaltknopf global (allgemein) verfügbar machen 740
Makro und Schaltknopf für Dokumente, welche auf einer bestimmten Vorlage basieren, verfügbar machen 741

Makro und Schaltknopf für ein bestimmtes Dokument verfügbar machen 741
Erscheinungsbild eines Schaltknopfes beeinflussen 742
Anmerkungen 742
Links zu diesem Thema 743

Der Umgang mit der Registrierdatenbank und dem Registriereditor 744
Bedeutung der Registrierdatenbank 744
Sicherheitsmassnahmen 744
Zugang zur Registrierungsdatenbank 744
Aufbau und Handhabung der Registrierungsdatenbank 744
Sichern und Wiederherstellen der Registrierdatenbank 745

Schlussbemerkungen 747
Programmversionen 747
Feedback 747
Hinweise für Hilfesuchende 747

Index

A

A5 Wie kann ich aus einem Word-Dokument ein Büchlein erzeugen?

Abbildungsverzeichnis Word findet bei der Generierung des Abbildungsverzeichnisses keine Einträge

Absätze Vom Umgang mit Zeilen und Seiten in MS Word

Absatzmarken Wie kann ich leere Absätze (Zeilen) aus einem Dokument entfernen?

Abschnitte drucken Wie kann ich einzelne Abschnitte oder einzelne Seiten in Word2000 drucken?

Absturz Hilfe! - Mein Word schmiert ab! (Alias: Die dreizehn Gebote, wenn Word spinnt!)

Access I (VBA) Von MS Word auf Access zugreifen

Access II (VBA) Eine Access-Tabelle in einer Word-Tabelle abbilden

Adressbuch I Wie kann ich auf die Adressen im Outlook Kontakte-Ordner zugreifen?

Adressbuch II Wie kann ich die Adressfelder und deren Reihenfolge, welche ich über den Befehl Einfügen Adresse in mein Dokument einfüge beeinflussen?

Adressbuch III Wieso kann ich nicht auf das Adressbuch von Outlook Express 4 oder 5 zugreifen?

Adresse einfügen Wie kann ich auf die Adressen im Outlook Kontakte-Ordner zugreifen?

Adressetiketten Wie kann ich beim Seriendruck - Adressetiketten ein Bedingungsfeld einbringen?

Adresslayout Wie kann ich die Adressfelder und deren Reihenfolge, welche ich über den Befehl Einfügen Adresse in mein Dokument einfüge beeinflussen?

ADO (VBA) Mittels «ADO» auf Access zugreifen

Ami Pro Zusätzliche Text- und Grafikfilter von Microsoft

Anrede Wie setze ich in einem Serienbrief eine bedingte Anrede?

ANSI I Wissenswertes über die Zeichensatztabellensuppe

ANSI II Wie lautet der ANSI-Wert zu den in Word gebräuchlichen Sonderzeichen, wie geschütztes Leerzeichen geschützter Trennstrich, etc.

Arbeitsmenü Wie kann ich obsolete Einträge zu Dateien im Arbeitsmenü oder in der Auflistung der zuletzt geöffneten Daten löschen?

ASCII I Wissenswertes über die Zeichensatztabellensuppe

ASCII II Wie lautet der ANSI-Wert zu den in Word gebräuchlichen Sonderzeichen, wie geschütztes Leerzeichen geschützter Trennstrich, etc.

AutoCAD Zusätzliche Text- und Grafikfilter von Microsoft?

AutoKorrektur I Wo werden Bausteine wie Formatvorlagen, AutoTexte, AutoKorrektur-Einträge und Symbolleisten gespeichert?

AutoKorrektur II Der Text in konventionellen Formulartextfeldern wird abgeschnitten oder verschluckt

AutoText I Wie kann ich eigene Untermenüs für AutoTexte erstellen?

AutoText II Wo werden Bausteine wie Formatvorlagen, AutoTexte, AutoKorrektur-Einträge und Symbolleisten gespeichert?

B - C

Baltisch — Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

Bausteine — Wo werden Bausteine wie Formatvorlagen, AutoTexte, AutoKorrektur-Einträge und Symbolleisten gespeichert?

Bedingungsfeld — Wie setze ich in einem Serienbrief eine bedingte Anrede?

Bedruckbarer Bereich — Die Seitenränder liegen ausserhalb des bedruckbaren Bereiches!

Befehle — Wie kann ich eine Liste der Word-Befehle erstellen?

Begriffe in Word — Verwechselbare Begriffe in MS Word

Benutzerwörterbuch — Benutzerwörterbücher?

Berechnung I — Wie kann ich Nullwerte in Berechnungen unterdrücken

Berechnung II — Wie kann ich auf fünf (Rappen) genau runden

Berechnung III — Datums- und Stundenberechnung mit Feldern, Textmarken, Formularfeldern und Tabellen

Beschriftungen — Word findet bei der Generierung des Abbildungsverzeichnisses keine Einträge

Betrag in Worten — Wie kann ich einen (Geld)-Betrag in Worte umwandeln?

Booklet — Wie kann ich aus einem Word-Dokument ein Büchlein erzeugen?

Büchlein — Wie kann ich aus einem Word-Dokument ein Büchlein erzeugen?

Combobox erstellen (VBA) — Symbolleistenschaltflächen und Menüleisten selbst gemacht

Comment Objekt (VBA) — Arbeiten mit dem Range-Objekt in MS Word

CorelDraw — Text- und Grafikformate, für welche Microsoft keine Filter zur Verfügung stellt

CreateObject (VBA) — Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen?

D

DAO (VBA) — Mittels «DAO» auf Access zugreifen

Datei auswählen (VBA) — Wie kann ich einen Dialog anzeigen, welcher es dem Benutzer erlaubt, einen Ordner bzw. eine Datei auszuwählen?

Datei Neu Dialog (VBA) — Der Word-Dialog «Datei - Neu...» soll statt grossen Symbolen kleine Symbole anzeigen

Datei Öffnen Dialog (VBA) — Wie kann ich einen Dialog anzeigen, welcher es dem Benutzer erlaubt, einen Ordner bzw. eine Datei auszuwählen?

Datei Öffnen/Speichern Dialog — Office2000: Wie kann ich die Word-Dialoge Datei - öffnen, Datei - speichern unter, Grafik - einfügen, etc. anpassen?

Dateieigenschaften — Wie lese oder setze ich mit VBA den Wert einer Dokumenteneigenschaft?

Dateikorruption — Der Umgang mit defekten Dokumenten

Dateiname in Kopfzeile — Wie kann ich kann ich den Dateiname eines Dokumentes mit Pfadangabe in die Kopf- oder Fusszeile setzen?

Datumsberechnung — Datums- und Stundenberechnung mit Feldern, Textmarken, Formularfeldern und Tabellen

Defekte Dokumente — Der Umgang mit defekten Dokumenten

Deinstallation — Wie kann Office 97 oder Office2000 möglichst vollständig von meinem Rechner entfernen?

Dokumenteigenschaften — Wie lese oder setze ich mit VBA den Wert einer Dokumenteneigenschaft?

Dokumentschutz — Rechtschreibprüfung und andere Funktionen stehen in einem teilweise geschützten Dokument nicht zur Verfügung

Dokumentvariablen — Vom Umgang mit Dokumentvariablen in MS Word (Einschliesslich Tool zur Verwaltung von Dokumentvariablen)

Dokumentvorlage (Definition) — Verwechselbare Begriffe in MS Word

DOS Dateien — Die Umlaute in einem DOS-Text werden beim Öffnen unter Word falsch dargestellt!

Drucken I — Wie kann ich einzelne Abschnitte oder einzelne Seiten in Word2000 drucken?

Drucken II — Die Seitenränder liegen ausserhalb des bedruckbaren Bereiches!

DXF — Zusätzliche Text- und Grafikfilter von Microsoft

E

Effekte — Es stehen keine Seitenrahmen (Zierrahmen) zur Verfügung!

Einfügemarke (VBA) — Wie kann ich in einem VBA-Programm (Makro) feststellen, auf welcher Seite sich die Einfügemarke befindet?

Einstellungen — Wo speichert MS Word seine Einstellungen (Umgebungsoptionen)?

Endnote Objekt (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Englische Feldnamen — Gegenüberstellung deutschsprachiger Feldnamen in Word 97 und englischsprachiger Feldnamen in Word2000

Enter Taste (VBA) — Das Betätigen der Enter-Taste in einem Formulartextfeld erzeugt einen Zeilenumbruch und verschiebt die Darstellung in meinem Formular

Entfernen von Office — Wie kann Office 97 oder Office2000 möglichst vollständig von meinem Rechner entfernen?

Eraser — Wie kann Office 97 oder Office2000 möglichst vollständig von meinem Rechner entfernen?

Ersetzen I (VBA) — Der Befehl Suchen/Ersetzen funktioniert nicht in Kopf- und Fusszeilen und auch nicht in Textfeldern

Ersetzen II — Wie kann ich Symbole suchen und ersetzen?

Ersetzen III — Wie kann ich beim wiederholten Ersetzen einer Zeichenfolge ausweisen, wie viele Ersetzungen getätigt wurden?

Etiketten I — Wie kann ich beim wiederholten Ersetzen einer Zeichenfolge ausweisen, wie viele Ersetzungen getätigt wurden?

Etiketten II — Wo speichert MS Word die Definitionen für benutzerdefinierte Etiketten?

Etiketten und Logos — Wie kann ich Etiketten mit einer Grafik (Logo) versehen?

Euro Symbol — Zusätzliche Schriften von Microsoft, welche das Euro-Symbol unterstützen

Excel I (VBA) — Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen?

Excel II (VBA) — Ein Excel-Tabellenblatt in einer Word-Tabelle abbilden

F

FaceID — Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen?

Faltmarke (VBA) — Wie bringe ich eine Faltmarke in ein Word-Dokument ein?

Farben bei Schriften — Word bietet mir nur 16 Schriftfarben an! Kann ich das ändern?

Feld (Definition) — Verwechselbare Begriffe in MS Word

Felder aktualisieren — Wie kann ich in allen Dokumententeile, also auch in Kopf- und Fusszeile sowie in Textfeldern die Felder aktualisieren?

Feldfunktion (benutzerdefiniert) — Kalenderwoche Feld-Funktion selbst gemacht!

Feldfunktionen — Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln?

Feldnamen — Gegenüberstellung deutschsprachiger Feldnamen in Word 97 und englischsprachiger Feldnamen in Word2000

Feldschalter — Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln?

Fenster — Wie kann ich verhindern, dass Word2000 für jedes Dokument, die Anwendung ein weiteres Mal startet?

Filter — Zusätzliche Text- und Grafikfilter von Microsoft

Find Objekt (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Footnote Objekt (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Formatvorlage (Definition) — Verwechselbare Begriffe in MS Word

Formatvorlagen I — Wo werden Bausteine wie Formatvorlagen, AutoTexte, AutoKorrektur-Einträge und Symbolleisten gespeichert?

Formatvorlagen II — Wie kann ich meine Formatvorlagen übersichtlich organisieren?

Formatvorlagenreferenz — Wie referenziere ich das jeweilige Kapitel in der Kopf- oder Fusszeile?

Formular (Definition) — Verwechselbare Begriffe in MS Word

Formulare I — Das Betätigen der Enter-Taste in einem Formulartextfeld erzeugt einen Zeilenumbruch und verschiebt die Darstellung in meinem Formular

Formulare II — Der Text in konventionellen Formulartextfeldern wird abgeschnitten oder verschluckt

Formularfeld (Definition) — Verwechselbare Begriffe in MS Word

Formularfelder (VBA) — Benutzerspezifische Standardwerte für konventionelle Formularfelder, Textmarkentext und UserForm-Controls

Formularschutz — Rechtschreibprüfung und andere Funktionen stehen in einem teilweise geschützten Dokument nicht zur Verfügung

Fortlaufende Nummer (VBA) — Wie kann ich in einem Word-Dokument oder in einem Word-Dokument eine fortlaufende Nummer vergeben?

Fortschrittsleiste (VBA) — Wie kann ich bei zeitlich langandauernden VBA-Prozessen eine Fortschrittsleiste darstellen?

Fremdsprachen — Wo kann ich Wörterbücher für Fremdsprachen beziehen?

Fünferrundung — Wie kann ich auf fünf (Rappen) genau runden

Fussnotenindex (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Fussnotentabulator (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Fussnotentext — Der Fussnotentext wird nicht auf der richtigen Seite gedruckt

Fussnotentrennlinie — Wie kann ich die Fussnotentrennlinie formatieren bzw. löschen?

Fussnotenzahl doppelt vergeben — Wie kann ich eine Fussnotenzahl doppelt vergeben?

G - J

Geschützte Dokumente — Rechtschreibprüfung und andere Funktionen stehen in einem teilweise geschützten Dokument nicht zur Verfügung

GetObject (VBA) — Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen?

Gitter — Wie kann ich eine Tabelle ohne Gitter erzeugen bzw. eine solche zum Standard küren?

Grafik I — Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Grafik II — Ein rotes Kreuz in einem Rechteck erscheint an der Stelle einer Grafik

Grafik festnageln — Objekte an einer bestimmten Stelle auf der Seite festnageln

Grafik Skalierung — Word vergisst die Skalierung von verknüpften Grafiken

Grafik und Etiketten — Wie kann ich Etiketten mit einer Grafik (Logo) versehen?

Grafik und Textfluss — Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Grafikfilter — Zusätzliche Text- und Grafikfilter von Microsoft

Grammatikprüfung — WD2000: Die Grammatikprüfung lässt sich nicht aktivieren.

Griechisch — Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

Grosse Dokumente — Soll man grosse und komplexe Dokumente (z.B. eine Diplomarbeit) in MS Word erstellen?

Grossschreibung — Das erste Wort nach einem Punkt wird immer mit einem Grossbuchstaben geschrieben

Header/Footer Objekt (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Hexadezimalzahl — Wie kann ich eine Dezimalzahl in eine hexadezimale Zahl umwandeln?

In Worten — Wie kann ich einen (Geld)-Betrag in Worte umwandeln?

Inhaltsverzeichnis (VBA) — Wie kann ich ein Inhaltsverzeichnis in ein anderes Word-Dokument übernehmen

InlineShape — Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Instanz — Wie kann ich verhindern, dass Word2000 für jedes Dokument, die Anwendung ein weiteres Mal startet?

Internet Explorer (VBA) — Wie kann ich aus anderen Anwendungen auf MS Word zugreifen?

K - M

Kalenderwoche (VBA) Datums- und Stundenberechnung mit Feldern, Textmarken, Formularfeldern und Tabellen

Kapitel drucken (VBA) Wie kann ich ein Kapitel aus einem gegliederten Dokument drucken?

Kapitelüberschrift Wie referenziere ich das jeweilige Kapitel in der Kopf- oder Fusszeile?

Kategorien in Outlook Das Outlook-Feld "Kategorie" steht in der Word-Seriendruckfunktion nicht zur Verfügung.

Keine Menüleiste MS Word startet nur noch ohne Menüleiste, Symbolleisten und Lineal!

Keine Schriften Es stehen keine oder nur zwei Schriften zur Verfügung

Kombinationsfeld erstellen (VBA) Symbolleistenschaltflächen und Menüleisten selbst gemacht

Kommentare umwandeln (VBA) Arbeiten mit dem Range-Objekt in MS Word

Kontakte I Wie kann ich auf die Adressen im Outlook Kontakte-Ordner zugreifen?

Kontakte II Von MS Word auf MS Outlook2000 zugreifen

Kontrollkästchen (Definition) Verwechselbare Begriffe in MS Word

Konverter Zusätzliche Text- und Grafikfilter von Microsoft?

Konvertieren von Dateien Wie kann ich in MS Word eine Datei unter einem fremdem Format öffnen oder abspeichern?

Konvertierung (Stapelverarbeitung) Wie kann ich eine Vielzahl von Dokumenten in ein anders Format bringen?

Konvertierung (VBA) VBA-technischer Umgang mit internen und externen Textkonverter

Kopf/Fusszeilen drehen Wie kann ich Kopf- und Fusszeilen um 90 Grad drehen?

Kopf/Fusszeilen (VBA) Arbeiten mit dem Range-Objekt in MS Word

Kopf/Fusszeilen löschen (VBA) Wie kann ich alle Kopf- und Fusszeilen eines Dokumentes löschen?

Kopfstehend Wie kann ich ein Word-Dokument spiegelverkehrt ausrucken?

Kyrillisch Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

Lange Dokumente Soll man grosse und komplexe Dokumente (z.B. eine Diplomarbeit) in MS Word erstellen?

Leerschläge beim Seriendruck Wie kann ich Leerschläge, welche aus leeren Felder resultieren, eliminieren?

Logos und Etiketten Wie kann ich Etiketten mit einer Grafik (Logo) versehen?

Lotus WordPro Text- und Grafikformate, für welche Microsoft keine Filter zur Verfügung stellt

Lückentext (VBA) Wie kann ich mit MS Word Lückentext erstellen und handlich zwischen Aufgabenstellung und Lösung umschalten?

Makro drucken (VBA) Wie kann ich alle VBA-Module einer Dokumentenvorlage oder eines Dokumentes ausdrucken?

Makro einrichten So bringen Sie ein Makro ein und verknüpfen es mit einer Schaltknopf auf einer Symbolleiste Ihrer Wahl

Markieren Wie kann ich mit MS Word nicht zusammenhängende Textstellen im Dokument gleichzeitig markieren?

MDB (VBA) Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen?

Mehrfachmarkierung Von MS Word auf Access zugreifen

Menü erstellen Symbolleistenschaltflächen und Menüleisten selbst gemacht

Menüleiste fehlt MS Word startet nur noch ohne Menüleiste, Symbolleisten und Lineal!

MS LineDraw MS Word und Schriften

N - Q

Nicht modaler Dialog I (VBA) _____ Wie kann ich bei zeitlich langandauernden VBA-Prozessen eine Fortschrittsleiste darstellen?

Nicht modaler Dialog II (VBA) _____ Office Assistent im ungebundenen (nicht modalen) Modus agieren lassen

Nullwerte unterdrücken _____ Wie kann ich Nullwerte in Berechnungen unterdrücken

Numerisches Bild_____ Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln?

Objekt einfügen_____ Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Office Assistent programmieren ____ Wie kann ich den Office Assistenten programmieren?

Office Automation (VBA)_____ Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen?

Office entfernen _____ Wie kann Office 97 oder Office2000 möglichst vollständig von meinem Rechner entfernen?

Office Journaling _____ Das Öffnen und Schliessen von Word-Dokumenten und anderen Office-Dateien dauert immer länger

Optionen _____ Wo speichert MS Word seine Einstellungen (Umgebungsoptionen)?

Ordner auflisten (VBA) _____ Wie kann ich die Verzeichnisstruktur einer Festplattenpartition, CD-ROM, Diskette oder eines Hauptverzeichnisses in einem Word-Dokument ausgeben?

Ordner auswählen (VBA) _____ Wie kann ich einen Dialog anzeigen, welcher es dem Benutzer erlaubt, einen Ordner bzw. eine Datei auszuwählen?

Outlook 97/98/2000_____ Wie kann ich auf die Adressen im Outlook Kontakte-Ordner zugreifen?

Outlook 2000 (VBA) _____ Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen?

Outlook Express _____ Wieso kann ich nicht auf das Adressbuch von Outlook Express 4 oder 5 zugreifen?

Outlook Kategorien _____ Das Outlook-Feld "Kategorie" steht in der Word-Seriendruckfunktion nicht zur Verfügung.

Passwortschutz_____ Ist der Passwortschutz für Word 97 und Word2000 Dokumente sicher?

Pfadname in Kopfzeile_____ Wie kann ich kann ich den Dateiname eines Dokumentes mit Pfadangabe in die Kopf- oder Fusszeile setzen?

Position bestimmen (VBA) _____ Wie kann ich in einem VBA-Programm (Makro) feststellen, auf welcher Seite sich die Einfügemarke befindet?

Positionsrahmen _____ Positionsrahmen versus Textfelder

Prolog _____ Prolog

R

Rahmen _____ Es stehen keine Seitenrahmen (Zierrahmen) zur Verfügung!

Ränder _____ Die Seitenränder liegen ausserhalb des bedruckbaren Bereiches!

Range Objekt (VBA) _____ Arbeiten mit dem Range-Objekt in MS Word

Raster _____ Wie kann ich eine Tabelle ohne Gitter erzeugen bzw. eine solche zum Standard küren?

Rechtschreibung I _____ Wo kann ich das Wörterbuch zur neuen deutschen Rechtschreibung beziehen?

Rechtschreibung II _____ Wo kann ich Wörterbücher für Fremdsprachen beziehen?

Rechtschreibung III (VBA) _____ Arbeiten mit dem Range-Objekt in MS Word

Registrierdatenbank _____ Der Umgang mit der Registrierdatenbank und dem Registriereditor

Römische Zahl _____ Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln?

Rotes Kreuz _____ Ein rotes Kreuz in einem Rechteck erscheint an der Stelle einer Grafik

Runden _____ Wie kann ich auf fünf (Rappen) genau runden

Russisch _____ Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

S

Satzanfang — Das erste Wort nach einem Punkt wird immer mit einem Grossbuchstaben geschrieben

Schalter — Welche Schalter (Switches) gibt es um die Word-Anwendung zu starten?

Schaltflächen — Symbolleistenschaltflächen und Menüleisten selbst gemacht

Schaltflächen zur Laufzeit (VBA) — Wie kann ich Schaltflächen einer UserForm dynamisch, während der Laufzeit erstellen?

Schaltknopf erstellen I — So bringen Sie ein Makro ein und verknüpfen es mit einer Schaltknopf auf einer Symbolleiste Ihrer Wahl

Schaltknopf erstellen II — Symbolleistenschaltflächen und Menüleisten selbst gemacht

Schliessen — Das Öffnen und Schliessen von Word-Dokumenten und anderen Office-Dateien dauert immer länger

Schliessen Kreuz I — Wie kann ich verhindern, dass Word2000 für jedes Dokument, die Anwendung ein weiteres Mal startet?

Schliessen Kreuz II (VBA) — Wie kann ich das Kreuz zum Schliessen einer UserForm in der Titelleiste unterdrücken?

Schrift — Wie lässt sich die Standartschrift für neue Word-Dokumente nachhaltig ändern?

Schriften — MS Word und Schriften

Schriften einbetten — MS Word und Schriften

Schriften eruieren (VBA) — Wie kann ich eruieren, welche Schriften in einem Dokument zur Anwendung kommen?

Schriften fehlen — Es stehen keine oder nur zwei Schriften zur Verfügung

Schriften installieren — MS Word und Schriften

Schriftenordner — MS Word und Schriften

Schriftenprobe (VBA) — Wie kann ich einen Beispieltext mit allen vorhandenen Schriften anzeigen oder ausdrucken?

Schriftfarben — Word bietet mir nur 16 Schriftfarben an. Kann ich das ändern?

Schutzverletzung — Hilfe! - Mein Word schmiert ab! (Alias: Die dreizehn Gebote, wenn Word spinnt!)

Seite bestimmen (VBA) — Wie kann ich in einem VBA-Programm (Makro) feststellen, auf welcher Seite sich die Einfügemarke befindet?

Seiten drucken — Wie kann ich einzelne Abschnitte oder einzelne Seiten in Word2000 drucken?

Seiten kopieren (VBA) — Vom Umgang mit Zeilen und Seiten in MS Word

Seiten löschen (VBA) — Vom Umgang mit Zeilen und Seiten in MS Word

Seiten — Vom Umgang mit Zeilen und Seiten in MS Word

Seitenrahmen — Es stehen keine Seitenrahmen (Zierrahmen) zur Verfügung!

Seitenränder — Die Seitenränder liegen ausserhalb des bedruckbaren Bereiches!

Seitenzahlen I — In der Kopf- oder Fusszeile erscheint die unsinnige Angabe Seite 7 von 1

Seitenzahlen II — Ich möchte die Seitenzahl auf der ersten Seite nicht anzeigen

Seitenzahlen III — Wie drucke ich in meinem Dokument Seitenzahlen nach der DIN?

Seitenzahlen IV — Word2000: In der Kopf- oder Fusszeile erscheinen die Angaben Seite 1 von 1, Seite 2 von 2, Seite 3 von 3, u.s.w.

Serienbrief — Wie setze ich in einem Serienbrief eine bedingte Anrede?

Serienbrief drucken — Wie kann ich einzelne Abschnitte aus einem Seriendruckdokument drucken?

Service Release — Wo kann ich den Office2000 Service-Release 1 beziehen?

SetRange Methode (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Shape — Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Silbentrennung — Wie lässt sich die automatische Silbentrennung nachhaltig aktivieren?

Sinnbilder — Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen?

Skalierung — Word vergisst die Skalierung von verknüpften Grafiken

Sonderzeichen Der Dialog Einfügen - Symbol... zeigt statt den Symbolen nur leere Quadrate an

Sonderzeichen kreieren Wie kann ich ein Symbol kreieren (z.B. ein x mit Überstrich, wie es in statistischen Berechnungen gebraucht wird)?

Sonderzeichen suchen/ersetzen Wie kann ich Symbole suchen und ersetzen?

Spalte (Definition) Verwechselbare Begriffe in MS Word

SpellingError Objekt (VBA) Arbeiten mit dem Range-Objekt in MS Word

Spiegelverkehrt Wie kann ich ein Word-Dokument spiegelverkehrt ausrucken?

Sprachunterstützung Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

Standardschrift Wie lässt sich die Standartschrift für neue Word-Dokumente nachhaltig ändern?

Standardwerte I Wo speichert MS Word seine Einstellungen (Umgebungsoptionen)?

Standardwerte II (VBA) Benutzerspezifische Standardwerte für konventionelle Formularfelder, Textmarkentext und UserForm-Controls

Standardwerte III (VBA) Werte von Textfeldern, Kontrollkästchen und Optionsfeldern einer UserForm speichern und beim nächsten Aufruf der UserForm einsetzen?

StarWriter Text- und Grafikformate, für welche Microsoft keine Filter zur Verfügung stellt

StoryRange (VBA) Arbeiten mit dem Range-Objekt in MS Word

Stundenberechnung Datums- und Stundenberechnung mit Feldern, Textmarken, Formularfeldern und Tabellen

Suchen/Ersetzen I (VBA) Der Befehl Suchen/Ersetzen funktioniert nicht in Kopf- und Fusszeilen und auch nicht in Textfeldern

Suchen/Ersetzen II Wie kann ich Symbole suchen und ersetzen?

Suchen/Ersetzen III Wie kann ich feststellen, wie oft ein bestimmtest Wort im Dokument vorkommt?

Suchen/Ersetzen IV (VBA) Arbeiten mit dem Range-Objekt in MS Word

Switches Welche Schalter (Switches) gibt es um die Word-Anwendung zu starten?

Symbole (Icons) Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen?

Symbole einfügen Der Dialog Einfügen - Symbol... zeigt statt den Symbolen nur leere Quadrate an

Symbole kreieren Wie kann ich ein Symbol kreieren (z.B. ein x mit Überstrich, wie es in statistischen Berechnungen gebraucht wird)?

Symbole suchen/ersetzen Wie kann ich Symbole suchen und ersetzen?

Symbolleisten I Wo werden Bausteine wie Formatvorlagen, AutoTexte, AutoKorrektur-Einträge und Symbolleisten gespeichert?

Symbolleisten II Symbolleistenschaltflächen und Menüleisten selbst gemacht

Symbolleisten schützen (VBA) Symbolleistenschaltflächen und Menüleisten selbst gemacht

Symbolleistenschaltflächen Symbolleistenschaltflächen und Menüleisten selbst gemacht

Symbolschriften MS Word und Schriften

T - U

Tabelle ohne Gitter — Wie kann ich eine Tabelle ohne Gitter erzeugen bzw. eine solche zum Standard küren?

Tabellen (VBA) — VBA-technischer Umgang mit Word-Tabellen

Tastaturtreiber — Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

Text drehen — Wie kann ich ein Word-Dokument spiegelverkehrt ausrucken?

Text konvertieren — Wie kann ich in MS Word eine Datei unter einem fremdem Format öffnen oder abspeichern?

Text markieren — Wie kann ich mit MS Word nicht zusammenhängende Textstellen im Dokument gleichzeitig markieren?

Textbausteine — Wo werden Bausteine wie Formatvorlagen, AutoTexte, AutoKorrektur-Einträge und Symbolleisten gespeichert?

Textfeld (Definition) — Verwechselbare Begriffe in MS Word

Textfelder I — Positionsrahmen versus Textfelder

Textfelder II (VBA) — Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Textfelder III (VBA) — Arbeiten mit dem Range-Objekt in MS Word

Textkonverter — Wie kann ich in MS Word eine Datei unter einem fremdem Format öffnen oder abspeichern?

Textmarken I — Wie lassen sich Textmarken in einem Word-Dokument bequem ansteuern?

Textmarken II (VBA) — VBA-technischer Umgang mit Textmarken

Textmarken III (VBA) — Benutzerspezifische Standardwerte für konventionelle Formularfelder, Textmarkentext und UserForm-Controls

Trennlinie — Wie kann ich die Fussnotentrennlinie formatieren bzw. löschen?

Türkisch — Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

Über den Text legen — Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Überschrift in Kopfzeile — Wie referenziere ich das jeweilige Kapitel in der Kopf- oder Fusszeile?

Umlaute und DOS Dateien — Die Umlaute in einem DOS-Text werden beim Öffnen unter Word falsch dargestellt!

Ungebundener Dialog I (VBA) — Wie kann ich bei zeitlich langandauernden VBA-Prozessen eine Fortschrittsleiste darstellen?

Ungebundener Dialog II (VBA) — Office Assistent im ungebundenen (nicht modalen) Modus agieren lassen

Ungültige Seite — Hilfe! - Mein Word schmiert ab! (Alias: Die dreizehn Gebote, wenn Word spinnt!)

Unicode I — Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen?

Unicode II — Wissenswertes über die Zeichensatztabellensuppe

UserForm I (VBA) — Benutzerspezifische Standardwerte für konventionelle Formularfelder, Textmarkentext und UserForm-Controls

UserForm II (VBA) — Werte von Textfeldern, Kontrollkästchen und Optionsfeldern einer UserForm speichern und beim nächsten Aufruf der UserForm einsetzen?

UserForm III (VBA) — Wie kann ich Schaltflächen einer UserForm dynamisch, während der Laufzeit erstellen?

UserForm schliessen (VBA) — Wie kann ich das Kreuz zum Schliessen einer UserForm in der Titelleiste unterdrücken?

V - Z

VBA Module drucken (VBA) — Wie kann ich alle VBA-Module einer Dokumentenvorlage oder eines Dokumentes ausdrucken?

Verschachtelte Tabellen (VBA) — VBA-technischer Umgang mit Word-Tabellen

Version ermitteln (VBA) — Wie lese oder setze ich mit VBA den Wert einer Dokumenteneigenschaft?

Verzeichnisbaum (VBA) — Wie kann ich die Verzeichnisstruktur einer Festplattenpartition, CD-ROM, Diskette oder eines Hauptverzeichnisses in einem Word-Dokument ausgeben?

Vorlage (Definition) — Verwechselbare Begriffe in MS Word

Währungstext — Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln?

Wasserzeichen — Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")

Wingdings — Der Dialog Einfügen - Symbol... zeigt statt den Symbolen nur leere Quadrate an

WOPR — Office2000: Wie kann ich die Word-Dialoge

Word für DOS — Wie kann ich in MS Word eine Datei unter einem fremdem Format öffnen oder abspeichern?

Word Instanz — Wie kann ich verhindern, dass Word2000 für jedes Dokument, die Anwendung ein weiteres Mal startet?

Word2000 I — Wie kann ich verhindern, dass Word2000 für jedes Dokument, die Anwendung ein weiteres Mal startet?

Word2000 II — Word2000: In der Kopf- oder Fusszeile erscheinen die Angaben Seite 1 von 1, Seite 2 von 2, Seite 3 von 3, u.s.w.

Word2000 III — Gegenüberstellung deutschsprachiger Feldnamen in Word 97 und englischsprachiger Feldnamen in Word2000

WordStar — Wie kann ich in MS Word eine Datei unter einem fremdem Format öffnen oder abspeichern?

Wörterbuch I — Wo kann ich das Wörterbuch zur neuen deutschen Rechtschreibung beziehen?

Wörterbuch II — Wo kann ich Wörterbücher für Fremdsprachen beziehen?

Worthäufigkeitstatistik (VBA) — Wie kann ich feststellen, wie oft ein bestimmtest Wort im Dokument vorkommt?

Zahlen formatieren — Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln?

Zahlen in Worten — Wie kann ich einen (Geld)-Betrag in Worte umwandeln?

Zeichencode I — Wissenswertes über die Zeichensatztabellensuppe

Zeichencode II — Wie lautet der ANSI-Wert zu den in Word gebräuchlichen Sonderzeichen, wie geschütztes Leerzeichen geschützter Trennstrich, etc.

Zeichencode III — Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen?

Zeile (Definition) — Verwechselbare Begriffe in MS Word

Zeilen — Vom Umgang mit Zeilen und Seiten in MS Word

Zeilenschaltung — Wie kann ich leere Absätze (Zeilen) aus einem Dokument entfernen?

Zierrahmen — Es stehen keine Seitenrahmen (Zierrahmen) zur Verfügung!

Zoom — Wie lässt sich die Zoom-Einstellung des Dokumentenfenster nachhaltig ändern?

Zuletzt geöffnete Dateien — Wie kann ich obsolete Einträge zu Dateien im Arbeitsmenü oder in der Auflistung der zuletzt geöffneten Daten löschen?

Zur 2. Ausgabe

◄ Zurück zum Hauptindex

Mit diesem Skript liegt Ihnen die **2. Gesamtausgabe** meiner **MS Word FAQ** (Frequently Answered Questions) im *PDF-Format* vor.

Bitte beachten Sie, dass dieses Werk keine kommerziellen Interessen verfolgt. Die Zeit, welche mir dafür zur Verfügung steht ist beschränkt und Perfektion wird nicht angestrebt. Das Skript ist unredigiert, bitte übersehen Sie grosszügig die sicher zahlreichen Schreib- und Tippfehler. Sachbezogene Fehler, Mängel, wichtige Ergänzungen und Reaktionen, senden Sie bitte an folgende E-Mail Adresse: René Probst.

Fachliche Fragen zu MS Word, werden von mir – Auf freiwilliger Basis - ausschliesslich in den von mir empfohlenen Foren und News-Gruppen beantwortet. Lesen Sie dazu auch die Schlussbemerkungen am Ende dieses Skriptes.

Word-FAQ im World Wide Web

Die aktuellste Version meiner MS Word-FAQ's finden Sie auch weiterhin online im Internet und zwar unter der Internet-Adresse (URL): http://mypage.bluewin.ch/reprobst. Das Skript wird dort aufgeteilt auf einzelne Kapitel publiziert und laufend erweitert und ergänzt. Periodisch kommen auch neue Artikel hinzu, welche Sie in dieser Gesamtausgabe noch nicht finden werden. Eine komfortable und sinnvolle Navigation ist in der Online-Version vorhanden. Im Internet stehen Ihnen auch die zahlreichen Querverweise und die sehr kontextspezifischen Links zu den technischen Publikationen von Microsoft und anderen Instanzen, welche im Zusammenhang mit MS Office relevant sind, per Mausklick zur Verfügung. Die Praxis hat gezeigt, dass der Server, welcher von *Bluewin* (eine 100%-ige Tochter des traditionellen Telekommunikationsunternehmens *Swisscom*) unterhalten wird, sehr gut erreichbar ist.

Nutzungsrechte

Das Skript im PDF-Format kann auch gedruckt werden. Um die Authentizität zu gewahren, kann es allerdings nicht verändert werden. Die Übernahme in andere Dateiformate unterstützt Adobe PDF willentlich nicht

! Die Weiterverbreitung dieser Publikation ist dann auch nur in der ursprünglichen Version gestattet. Alle Rechte bleiben beim Verfasser.

Was Sie erwartet

- Das Skript, welches Ihnen vorliegt enthält unter anderem:
- ca. 70 Artikel rund um MS Word und Word-VBA
- Ein umfangreiches Stichwortverzeichnis
- Knapp 100 Abbildungen
- Über 300 VBA-Code Beispiele in fast allen Schwierigkeitsstufen
- Unzählige Hyperlinks innerhalb des Skriptes
- Über 200 kontextbezogene Hyperlinks zu ergänzenden Artikeln im World Wide Web
- Über 700 Seiten Volumen

VBA-Code Beispiele

Die hier publizierten Code-Beispiele wurden teilweise unter Word 97, vermehrt aber auch unter Word2000 entwickelt und getestet. Wenn Sie VBA-Code heraus kopieren, dann bedenken Sie, dass dies einfacher geht, wenn Sie meine Home Page im Internet besuchen. Dort liegen die Code-Beispiele im HTML-Format vor und können deshalb sehr einfach mit Kopieren/Einfügen übernommen werden. In diesem Falle haben Sie auch Gewähr, dass Ihnen die neuste Version der Beispiele vorlieget.

Und nun wünsche ich Ihnen viel Spass und Erfolg mit dem dargebotenen Skript.

René Probst, März 2001

Prolog

◄ Zurück zum Hauptindex

Schwerpunkt ist Word 97 und Word2000. Wenn nichts anderes erwähnt ist, lassen sich die Ratschläge auf beide Versionen anwenden Zumindest das Datei-Format ist ja bei beiden Versionen über weite Strecken das selbe geblieben.

Obwohl wir schon zwei Service Releases von Office 97 (bzw. ein Service Release und ein Service Pack von Office2000) hinter uns bringen mussten, welche sich im Falle von Office 97 auf manchen Rechnern nur nach dem Studium von einem halben Duzend technische Abhandlungen von Microsoft aufzuspielen liessen, haben die Fehler und Unzulänglichkeiten von Word 97 oder Word2000 nicht wirklich entschieden abgenommen. Viele Probleme sind längstens bekannt und zu einem guten Teil auch von Microsoft dokumentiert, Abhilfe gibt es aber, wenn dann überhaupt, oft nur in Form eines sogenannten Work-around's. Oft geht aber viel Zeit verloren, bis man als Anwender auf einen solchen Rundherumlauf stösst. Zuerst zweifelt der Benutzer an sich, weil Textverarbeitung ein erstaunlich komplexes Gebiet ist und Word wegen der zahlreichen Bugs und Unzulänglichkeiten den Benutzer verunsichert. Als nächstes zweifelt der Anwender an der Rechnerumgebung und erst dann geht er davon aus, dass das Problem in der Word-Umgebung zu suchen ist. Diese Reihenfolge, welche auch gerne von Leuten, welche der Firma Microsoft nahe stehen, propagiert wird, ist aber nicht immer sehr effizient. Erfahrungsgemäss ist die Hardware und was mit ihr zusammenhängt, erstaunlich stabil. Die Software ist ihr schon immer hinten her gerannt. Tja, wenn sich Grafikkarten nur zur Wehr setzen könnten! Manchmal wünschte ich mir, die Firma Microsoft würde in eigener Regie Grafikkarten herstellen, dann wäre wenigstens dieser Hickhack, welcher auch munter von Microsoft betrieben wird, sinnlos bzw. Microsoft Mitarbeiter und Mitarbeiterinnen könnten ihn intern ausmachen.

Einiges deutet darauf hin, dass sich in der neusten Version von Word, Verbesserungen und neu generierte Probleme etwa die Waage halten, wobei zu berücksichtigen ist, dass der Funktionsumfang im neusten Kind von Microsoft, wieder etwas zugenommen hat. Die Vernetzung hat ebenfalls zugenommen, was bei mir ein mummeiliges Gefühl zurück lässt. Die ersten Erfahrungen decken sich leider mit früheren Versionenwechseln: die Summe der Probleme unter dem Strich ist immer etwa gleich gross geblieben. Auch habe ich nicht den Eindruck, dass bei der Entwicklung von Word2000 die Verbesserung der Zuverlässigkeit das Schwerpunksthema wahr.

Obwohl es sich bei Word2000 schon um die sechste Generation eines Textprozessors à la Microsoft handelt (berücksichtig man auch MS-DOS und Macintosh dann sind es dreizehn Versionen), ist das Produkt noch weit davon entfernt, wirklich zuverlässig zu funktionieren. Vor allem zu bemängeln ist, dass die Fehlersuche oft immer noch einem Spiessrutenlauf gleich kommt. In vielen Betrieben, gehen somit unzählige Stunden teurer Arbeitszeit verloren. Für mich oft unverständlich ist die Tatsache, dass Fehler über Jahre und Generationen von Word-Versionen zwar auch Microsoft bekannt sind, nachgebessert werden Sie aber trotzdem nicht. Hier frage ich mich, wo eigentlich der Berufsstolz jener Leute beleibt, welche Word entwickeln bzw. die Entwicklung beaufsichtigen und koordinieren. Üblicherweise tauscht man bei einem Haus, welches auf einem wackligen Fundament steht und auch sonst in einem schlechten Zustand ist, keinen Ziegel mehr aus, weil zu befürchten ist, dass es bei der kleinsten Veränderung wie ein Kartenhaus in sich zusammen bricht.

Völlig unverständlich für mich. ist die Tatsache, dass Microsoft immer noch nicht damit begonnen hat, die Fehlermeldungen, welche die hauseigenen Produkte von sich geben, zu enummerieren. Diese Arbeit wurde im Grossrechnerbereich in den letzten zehn Jahren systematisch angepackt. Jede Fehlermeldung hat dort eine eindeutige Meldungsnummer und die Meldungen wurden systematisch katalogisiert. Diese Arbeit hat zum Teil zu fast schon sensationellen Ergebnissen geführt. Heute findet man die Dokumentation zu einer Fehlermeldung in diesem Bereich auf Knopfdruck, ganz gleichgültig in welcher Sprache diese ausgegeben wurde. Vorbei sind also die mühseligen Zeiten, wo das Fachpersonal, die oft schlecht vom Englischen ins Deutsche übersetzten Meldungen wieder zurück ins Englische übersetzen müssen, um überhaupt einen Anhaltspunkt zu finden. Die systematische Bearbeitung von Meldungen hat aber noch weit gewichtigere Vorteile gebracht. Es hat sich zu einem unverzichtbaren Führungsinstrument gemausert. Heute ist es ein Einfaches heraus zu finden, welche Anwendungen oder Komponenten besonders viele Fehler hervorrufen. Damit ist ein gezieltes Vorgehen, um die störanfälligen Komponenten zu verbessern, erst möglich. In der Industrie werden heute dann auch rund 96% - 98% der erscheinenden Meldungen automatisch abgehandelt. Solche Automatismen basieren auf ein ganzes Arsenal von Möglichkeiten, welche sich nach einer gründlichen Analy-

se des Materials ergeben. Ein guter Prozentsatz der Meldungen, ist nämlich schlicht überflüssig. Andere Meldungen wiederum, sind nur in einem bestimmten Kontext relevant. Wiederum ein gutes Stück des Kuchens, kann mit selbstauslösenden Routinen bereinigt werden. Andere wiederum lassen zu mindest einen direkten und gezielten Griff auf die *Knowledge Base* zu. Was dann noch übrig bleibt, wird in einem sogenannten *Message Learning Tool* gesammelt und die verantwortliche Entwicklungsabteilung ist angehalten, sich für den nächsten *Software Release* etwas einfallen zu lassen. Wie schon angedeutet, handelt es sich bei der systematischen Erfassung von Fehlermeldung mit dem Endziel Automation um ein langjähriges Projekt. Ein Projekt, welches auch die Fähigkeit voraussetzt, selbstkritisch in den Spiegel zu schauen und sich mit den Fehlern zu identifizieren. Eine Betriebs- oder Anwendungssoftware im Grossrechnerbereich, welche heute noch solche unsystematisch aufgebaute Verse von sich gibt, wie man dies von Microsoft Produkten leider gewohnt ist, hat auf dem Markt überhaupt keine Chancen mehr.

Wenn ich im deutschen PC-Magazin **c't** in einem Interview mit Bill Gates lese, die Firma Microsoft möchte in Zukunft intelligent und wo möglich automatisch auf Fehler reagieren, dann weiss ich nicht, ob ich loslachen oder heulen soll. Wo bleiben die fundamentalen Ansätze für ein solches langjähriges Projekt?. Wie lange müssen wir Kunden uns noch gedulden, bis wir in den Genuss solcher Mechanismen kommen? Auf einen Nenner gebracht: schwatzt nicht! *Just do it!*. Holt die zehn Jahre, welche Ihr verschlafen habt nach! Der Prinz schaut nur in den Märchen vorbei und auch das soll hundert Jahre dauern!

Befasst man sich längere Zeit intensiv mit der Problematik von Word, dann erkennt man früher oder später, dass die funktionellen Schwächen dieses Produkts endlich sind. Damit meine ich, dass sich nach einer gewissen Zeit die Fragen, welche beim Benutzer auftauchen, wiederholen. Nicht zuletzt deshalb erachte ich, ein Skript wie dieses als wertvoll.

Schlimmer sieht die Sache bezüglich der Vermeidung von unkontrollierten Abstürzen, in Fachkreisen **IPF's** (*Invalid Page Fault* = Fehler in ungültiger Seite) und **GPF's** (*General Protection Fault* = Allgemeine Schutzverletzung) aus. Diese Fehler tauchen in unregelmässigen Abständen in immer neuem Kontext und mit verschiedenen Schattierungen auf. Man könnte fast zum Schluss kommen, dass diese sich selbst mutieren könnten. Kommt hinzu, dass die PC-Betriebssysteme aus dem selben Software-Haus, in den wenigsten Fällen in der Lage sind, nach einem derartigen Unfall wieder sauber aufzuräumen. Die Task bleibt in der Regel im Hintergrund aktiv, der belegte Adressraum (Arbeitsspeicher) wird nicht frei gegeben. Dies wären aber Aufgaben eines soliden Betriebssystems, welches auch in solchen Situationen Oberhand behalten sollte. Die Anwendung selber ist dafür ja nicht mehr fähig, da sie sich bereits verabschiedet hat. Bleibt uns Benutzern und Benutzerinnen also nur noch, Windows neu zu starten und zu hoffen, dass der Datenverlust nicht allzu gross ist und dass das unkontrollierte Verhalten nicht weitere Schäden auf dem sorgsam gepflegten System angerichtet hat. Hier, sollte sich Microsoft langsam etwas einfallen lassen, wollen Sie nicht in den nächsten vier bis sieben Jahren von einem Softwarehersteller überflügelt werden. Ich weiss nicht im Detail was die Zukunft hier bringen wird. Ich bin aber überzeugt, dass die Leute welche beweisen werden, dass man auch stabilere Anwendungen entwickeln kann, bereits geboren sind und dereinst Microsoft vorführen werden.

Erfahrungsgemäss wird dies ein heilsamer Schock für eine Firma sein, welche heute noch Oberhand hat. Spätestens dann, werden wir auch von Microsoft mit qualitativ besseren Produkten versorgt werden. Die Zuverlässigkeit der Word-Anwendung entspricht heute ungefähr dem Stand eines Personenkraftfahrzeugs in den 30-er oder 40-er Jahren. Und in den siebziger Jahren, als wir glaubten, ein Auto zu haben, welches nur noch punktuell zu verbessern ist, wurden die japanischen Ingenieure gross und bewiesen uns, dass es auch noch günstiger und vor allem qualitativ besser geht.

Dieses Skript soll aber nicht nur eine Aneinanderreihung von Fehlern in der Word-Anwendung bieten, sondern auch Tipps geben, wie sich Word flexibler nutzen lässt. Kapitel, wie der Abschnitt welches sich mit Office Automation auseinander setzt, geben dafür ein Beispiel ab. Die objektorientierte Programmiersprache von MS Office, nämlich VBA, ist eine Quelle, welche heutzutage noch viel zu wenig genutzt wird. In diesem Skript soll deshalb auch dieser Bereich nicht zu kurz kommen.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Prolog.htm
René Probst, Oktober 1999

Änderungen
- 26.Juni 2000 - Office2000

Verwechselbare Begriffe in MS Word

◄ Zurück zum Hauptindex

Problem

> Auch ich habe Spass daran, über Problemstellungen in MS Word zu diskutieren. Schwierig, unnötig zeitraubend und missverständlich wird es allerdings, wenn mit falschen oder schwammigen Bergriffen operiert wird. Dieses Skript soll dazu beitragen, Begriffe, welche sich im Umfeld von MS Word als missverständlich erweisen, zu entwirren und den Sinn für klare Begriffe schärfen.

Diese Seite

- ▼ Der Begriff MS Word
- ▼ Der Begriff Zeile bzw. Zeilenumbruch
- ▼ Der Begriff Spalte
- ▼ Der Begriff Vorlage
- ▼ Die Begriffe Formular, Formularfelder und Feld
- ▼ Der Begriff Textfeld
- ▼ Der Begriff Grafik bzw. Objekt
- ▼ Der Begriff Textmarke
- ▼ Der Begriff Schaltfläche, Knopf oder ähnliche Begriffe
- ▼ Der Begriff Kontrollkästchen bzw. CheckBox
- ▼ Der Begriff Listenfeld bzw. ListBox
- ▼ Der Begriff Kombinationsfeld oder ComboBox

Der Begriff *MS Word*

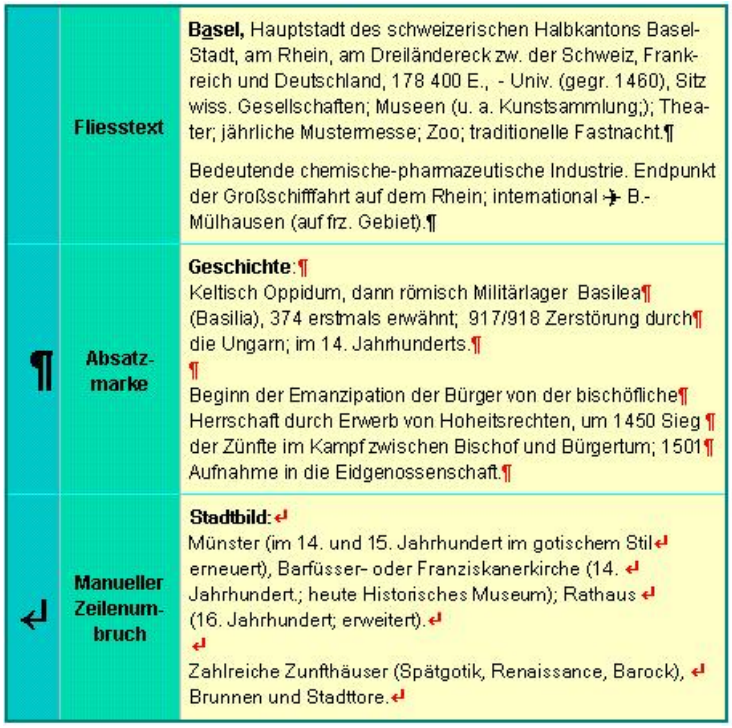
▲ Zurück zum Anfang

Zu fast allen Anfragen im Zusammenhang mit MS Word gehört es sich, die Word-Version anzugeben. Folgende Word-Versionen sind noch verbreitet (in der Reihenfolge ihrer Bedeutung):

- MS Word für Windows
  - Word2000
  - Word 97
  - Word 95 (Auch als Word 7.0 bekannt)
  - Word 6.x
- MS Word für Macintosh
  - Word 98
  - Word2001

Der Begriff *Zeile* bzw. *Zeilenumbruch*

▲ Zurück zum Anfang



Word kennt verschiedene Arten von Zeilenumbrüche. Im Zusammenhang mit Fragen zu Zeilen sollten Sie also unbedingt angeben, um welche Art von Zeilenumbruch es sich handelt. Das Verhalten kann nämlich in der Tat sehr unterschiedlich sein. Sehen Sie sich dazu auch dir Grafik auf dieser Seite an.

- Automatische (zufällige) Zeilenumbrüche als Folge von Fliesstext.
- Manuelle Zeilenumbrüche (erzeugt mittels **[Umschalt + Enter]**).
- Ein Absatzwechsel (erzeugt mittels **[Enter]**) impliziert selbstverständlich ebenfalls einen forcierten Zeilenumbruch. In der Regel hat man das Absatzformat so eingerichtet, dass sich ein neuer Absatz optisch mit zusätzlichem Zwischenstand (Durchschuss) vom vorhergehenden Absatz absetzt.

Daneben gibt es selbstverständlich auch noch Tabellenzeilen. Das ist dann aber was ganz anderes.

Der Begriff *Spalte*

▲ Zurück zum Anfang

Beim Begriff Spalten unterscheidet man zwischen **Tabellenspalten** (englisch *Columns*) und **Zeitungsspalten** (englisch Text *Columns*). Das erste sind Elemente von Word-Tabellen, das zweite erreichen Sie, wenn sie vom Word-Hauptmenü den Befehl *Format - Spalten...* ausgeben.

Der Begriff *Vorlage*

▲ Zurück zum Anfang

Beim einer **Vorlage** handelt es sich um eine sehr zentrale Sache in MS Word. Meistens meint der Benutzer damit eine **Dokumentvorlage** oder aber eine **Formatvorlage**. Obwohl es sich dabei um zwei ganz verschiedene Dinge handelt, werden diese Begriffe durcheinander gebracht.

Wird die Sache aus dem Zusammenhang nicht klar, dann gehe ich beim Begriff *Vorlage* davon aus, dass der Benutzer damit eigentlich eine Dokumentvorlage meint.

Wesentlich schöner sind diese Begriffe im englischen ausformuliert. Die Dokumentvorlage heisst dort *Template* und die Formatvorlage *Style*.

Im folgenden wird erkläret, was eine Dokumentvorlage bzw. eine Formatvorlage in MS Word bedeutet und was der Sinn dieser Komponenten ist.

Dokumentvorlage

Eine Dokumentvorlage ist ein Muster (in Form einer Datei), welches bei der Erstellung eines neuen Dokumentes auf das neue Dokument übertragen wird. In MS Word ist es so eingereichtet, dass jedes Dokument auf genau einer Dokumentvorlage basiert. Wählen Sie als Benutzer keine spezielle Dokumentvorlage aus, kommt die Standardvorlage mit dem Namen *Normal.dot* zum Zuge.

Dokumentvorlagen sind Dateien mit der Dateierweiterung ***.dot**. Sie können Standardtext, Grafiken, Formatvorlagen, AutoTexte und anderes beinhalten. Diese Elemente werden dann auf das neue Dokument 1:1 übertragen bzw., stehen solange zur Disposition, bis das Dokument und somit auch die dazugehörige Dokumentvorlage geschlossen wird.

Formatvorlagen

Formatvorlagen dienen der (einheitlichen) Formatierung von Textteilen. Man unterscheidet zwischen Formatvorlagen, welche **Zeichen-orientiert** sind und anderen welche **Absatz-orientiert** sind. Dank der ersteren Kategorie ist es möglich, einzelne Zeichen oder Wörter effizient und einheitlich zu formatieren. Die letzteren greifen vor allem in den Textfluss des Dokumentes ein. Sie dienen z.B. dazu die Abstände zwischen Zeilen bzw. zwischen Absätzen zu regeln. Über eine Absatzformatvorlage legt man auch die Textausrichtung und vieles mehr, fest. Das Formatierung eines Schriftstückes sollte vorzugsweise indirekt über solche Formatvorlagen geschehen, statt direkt im Text einzugreifen. Formatvorlagen sind Elemente, welche entwerde in einer Dokumentvorlage oder im Dokument selbst gespeichert sind. Formatvorlagen dieses Typus haben in der Regel einen höheren Stellenwert, als die Formatvorlagen des Typus Zeichen.

Zeichenformatvorlage

Eine Formatvorlage vom Typus Zeichen ermöglicht die effiziente und einheitliche Formatierung von einzelnen Zeichen oder Wörter. Dabei wird nicht in den Textfluss eingegriffen. Oft wird dieser Typus von Formatvorlagen dazu verwendet, um kürzere Textstellen optisch - einheitlich hervorzuheben. Die Zeichenformatierung kann übrigens auch über eine Absatz-Formatvorlage geschehen.

Absatzformatvorlage

Eine Formatvorlage vom Typus Absatz greift in der Regel in den Textfluss ein. Sie dient dafür die Abstände zwischen Zeilen und zwischen einer Anhäufung von Absätzen einheitlich festzulegen. Auch die Ausrichtung des Textes wird mit Vorteil in einer solchen Formatvorlage festgelegt. Absatz-Formatvorlagen mit Namen wie Titel, *Überschrift 1*, *Aufzählung 1*, etc. können auch als Strukturelemente des Textes angesehen werden.

Die Begriffe *Formular*, *Formularfelder* und *Feld*

▲ Zurück zum Anfang

Formular ist ein sehr schwammiger Begriff. Sie müssen hier unbedingt präziser formulieren. Bei einem Formular denke ich an einen der folgenden Komponenten:

- Eine **UserForm** *(VBA-Editor - Einfügen - UserForm*).
- Ein Word-Dokument, welches mit **konventionellen Formularfeldern** aus der Symbolleiste mit der Bezeichnung **Formular** bestückt ist.
- Ein Word-Dokument, welches mit **ActiveX-Komponenten** aus der Symbolleiste mit der Bezeichnung **Steuerelement-Toolbox** bestückt ist.

Beim Begriff **Formularfeld** denke ich an einen der folgenden Elemente:

- Ein **Textfeld** oder ähnliches auf einer **UserForm**.
- Ein **konventionelles Formularfeld**, welches über die Symbolleiste mit der Bezeichnung **Formular** in das Word-Dokument eingebracht wurde.
- Eine **ActiveX-Komponenten** aus der Symbolleiste mit der Bezeichnung **Steuerelement-Toolbox** Welches direkt in das Dokument eingebracht wurde.
- Ein **Textmarkenfeld**, welches über den Befehl *Einfügen - Textmarke* bei erweiterter Markierung in das Dokument eingebracht wurde.
- Beim Begriff **Felder** denke ich in erster Linie an Felder, welche über den Befehl *Einfügen - Feld...* in das Dokument eingebracht wurde. Die selben Felder können aber noch auf andere Arten in das Dokument gelangen explizit oder implizit).

Der Begriff *Textfeld*

▲ Zurück zum Anfang

Beim Begriff **Textfeld** denke ich an einen der folgenden Elemente:

- Ein **Textfeld** welches über die Symbolleiste mit der Bezeichnung **Zeichnen** in das Dokument eingefügt wurde.
- Ein **Textformularfeld**, welches über die Symbolleiste mit der Bezeichnung **Formular** in das Word-Dokument eingebracht wurde.
- Eine **ActiveX-Komponenten** (Textfeld) aus der Symbolleiste mit der Bezeichnung **Steuerelement-Toolbox** Welches direkt in das Dokument eingebracht wurde.
- Ein **Textmarkenfeld**, welches über den Befehl ***Einfügen - Textmarke*** bei erweiterter Markierung in das Dokument eingebracht wurde.
- Ein **Textfeld**, welches auf einer **UserForm** platziert wurde.

Der Begriff *Grafik* bzw. *Objekt*

▲ Zurück zum Anfang

Das selbe Bild – einmal als «Shape»...

Wenn Sie von Grafiken oder ähnlichen Objekten im Dokument sprechen. Dann dürfen Sie es nicht versäumen, anzugeben, auf welche Art und Weise diese Elemente in das Dokument eingebracht wurden.

MS Word unterscheidet nämlich strikte zwischen sogenannten **Shapes** (ungebundene bzw. schwebende Objekte) und **InlineShapes** (an den Text gebundene Objekte).

Eine klare Unterscheidung ist deshalb nötig, da das Verhalten von *Shapes* bzw. I*nlineShapes* sehr unterschiedlich ist. Viele Methoden, welche für *Shapes* zur Verfügung stehen, sind auf *InlineShapes* nicht anwendbar und umgekehrt.

... und als «InlineShape».

Achten Sie auf die Unterschiedliche Markierung.

«Shapes» und «InlineShape» sehen gleich aus, zeigen aber ein unterschiedliches Verhalten.

Auch Word-VBA unterscheidet strikte zwischen *Shapes* und *InlineShapes*, handelt es sich dort doch um unterschiedlich Objekte im Sinne des Word-Objekt-Modells. Die Frage, ob ein solches Objekt als *Shape* oder als *InlineShape* eingebracht wurde, ist oft entscheidender als die Frage, ob es sich nun z.B. um eine Grafik, ein eingebettetes Excel-Diagramm oder eine im Dokument eingefügte *ActiveX*-Komponente handelt.

Um zu erkennen, ob es sich beim vorliegenden Element um eine *Shape* oder aber um eine *InlineShape* handelt, markieren Sie das Objekt. Am Aussehen der Markierung ist dann eindeutig erkennbar, ob eine *Shape* oder eine I*nlineShape* vorliegt. Vergleichen Sie dazu die beiden Abbildungen in diesem Skript.

Wenn Sie mehr über den Unterschied zwischen *Shapes* und I*nlineShapes* erfahren möchten, dann beachten Sie bitte folgendes Skript:

► **Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")**

Der Begriff *Textmarke*

▲ Zurück zum Anfang

Im Zusammenhang mit **Textmarken** ist es sehr oft wichtig zu erfahren, ob es sich um eine Textmarke handelt, welche ausschliesslich eine Position markiert (**offene Textmarke**) oder um eine **Textmarkenfeld** (geschlossene Textmarke), welche auch ein Bereich darstellt. Im folgenden wird dieser Unterschied nochmals detaillierter erklärt:

Arten von Textmarken und ihr Verhalten

Was viele zu wenig beachten, Word kennt eigentlich zwei Arten von Textmarken, nämlich offene und geschlossene. Um dies zu erkennen, machen wir folgenden Test:

Offene Textmarken

Damit Textmarken überhaupt am Bildschirm angezeigt werden, sollten Sie zuerst die Einstellung ***Extras - Optionen - Ansicht - Anzeigen - Textmarken*** überprüfen.

Setzen sie die Einfügemarke im Dokument an die gewünschte Stelle, und wählen Sie **ohne dass etwas im Dokument markiert wäre**, den Dialog ***Einfügen - Textmarke...*** und vergeben Sie der Marke einen Namen. Das Resultat ist eine **offene Textmarke**. Sie wird wie folgt angezeigt: `[`. Mit diesem Vorgehen, haben Sie letztlich eine **Position** im Dokument definiert.

Textmarkenfelder (Geschlossene Textmarken)

Als nächstes markieren Sie einen kurzen Text z.B. ein Wort im Dokument. und wählen dann wieder den Dialog ***Einfügen - Textmarke...*** dann vergeben Sie der Marke einen Namen. Das Resultat ist eine **geschlossene Textmarke**. Sie wird wie folgt angezeigt: `[ ]`. Sie haben letztlich einen **Bereich** definiert.

Der Begriff *Schaltfläche*, *Knopf* oder ähnliche Begriffe

▲ Zurück zum Anfang

MS Word kennt eine Vielzahl von verschiedenen Schaltflächen um den Umgang mit der Anwendung für den Benutzer einfacher zu gestallten. Damit meine ich jetzt nicht nur, dass es z.B. für eine *UserForm* mindestens ein halbes Duzend Arten von Schaltflächen gibt; ich spreche viel mehr die Kategorien an. Deshalb sollten Sie bei Anfrage bezüglich dieser Elemente präzise sein. Knöpfe, welche beim Anklicken etwas auslösen, können sich im Dokument selbst (eher die Ausnahme), auf einer der Symbolleisten (inklusive Menüleiste) oder auf einer *UserForm* befinden. Sollten Sie den richtigen Begriff nicht kennen, dann können Sie auch kurz beschreiben, wie der Schaltknopf eingefügt wurde. Damit wird die Sache nämlich auch verständlich und eindeutig.
Die folgende Auflistung soll Ihnen helfen, genau jenen Begriff zu finden, welcher auf Ihre Situation zutrifft.

- Schaltflächen im **Dokument** selbst
  - Befehlsschaltflächen direkt im Word-Dokument eingefügt über die Symbolleiste **Stereurelement-Toolbox**. Der englische Begriff dazu lautet *CommandButton,* wobei zu sagen ist, dass das entsprechende Element auf einer UserForm genau gleich lauten.
- Schaltflächen (Symbole) auf einer **Symbolleiste**
  - Der englische und Office-interne Begriff dazu lautet **CommandBarButton**.
- Schaltflächen auf einer **UserForm**
  - Befehlsschaltflächen auf einer UserForm. Der englische Begriff dazu lautet *CommandButton,* wobei zu sagen ist, dass die entsprechenden *ActiveX*-Komponenten eingefügt im Dokument über die Symbolleiste mit der Bezeichnung **Stereurelement-Toolbox** genau gleich lauten.

Der Begriff *Kontrollkästchen* bzw. *CheckBox*

▲ Zurück zum Anfang

Kontrollkästchen bzw. englisch *CheckBoxes* können in MS Word an folgenden drei Orten auftreten:

- Auf einer **UserForm**
- Eingebettet als ***ActiveX*-Komponente** in einem Dokument. In diesem Fall wurde die Schaltfläche über die Symbolleiste mit der Bezeichnung **Steuerelement-Toolbox** eingebracht.
- Eingebettet als **konventionelles Formularfeld** in einem Dokument. In diesem Fall wurde die Schaltfläche über die Symbolleiste mit der Bezeichnung **Formular** eingebracht.

Sollten Sie Hilfe anfordern, ist es unerlässlich klar festzuhalten, welche Art von Kontrollkästchen Sie nun ansprechen.

Der Begriff *Listenfeld* bzw. *ListBox*

▲ Zurück zum Anfang

Listenfelder bzw. englisch *ListBoxes* können in MS Word an folgenden zwei Orten auftreten:

- Auf einer **UserForm**
- Eingebettet als ***ActiveX*-Komponente** in einem Dokument. In diesem Fall wurde die Schaltfläche über die Symbolleiste mit der Bezeichnung **Steuerelement-Toolbox** eingebracht.

Sollten Sie Hilfe anfordern, ist es unerlässlich klar festzuhalten, welche Art von Listenfeld Sie nun ansprechen. Bei Word-Formularen (Dokumente, welche mit Objekten aus der Symbolleiste mit der Bezeichnung Formular bestückt sind), gibt es noch **Dropdown-Formularfelder**. Das ist dann wieder etwas anderes.

Der Begriff *Kombinationsfeld* oder *ComboBox*

▲ Zurück zum Anfang

Kombinationsfelder bzw. englisch *ComboBoxes* können in MS-Office an folgenden drei Orten auftreten:

- Auf einer **UserForm**
- Eingebettet als ***ActiveX*-Komponente** in einem Dokument. In diesem Fall wurde die Schaltfläche über die Symbolleiste mit der Bezeichnung **Steuerelement-Toolbox** eingebracht.
- Auf einer **Symbolleiste**.

Sollten Sie Hilfe anfordern, ist es unerlässlich klar festzuhalten, welches Kombinationsfeld Sie nun ansprechen.

Bei Word-Formularen (Dokument welches mit Objekten aus der Symbolleiste mit der Bezeichnung **Formular** bestückt sind, gibt es noch **Dropdown-Formularfelder**. Das ist dann wieder etwas anderes.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Begriffe.htm

René Probst, Januar 2001

Änderungen

- --- nil ---

Soll man grosse und komplexe Dokumente (z.B. eine Diplomarbeit) in MS Word erstellen?

◄ Zurück zum Hauptindex

Problem

> **Langsam aber unausweichlich, beginnt auch für mich der Ernst des Lebens. In ein paar Monaten muss ich meine Dissertation in Angriff nehmen und dann binnen nützlicher Zeit zu Ende führen. Ich schätze, dass meine Dissertation so um die 80 bis 200 Seiten umfassen wird. Das Dokument erfordert komplexe Formatierung, es soll auch Grafiken enthalten und die Professoren und Professorinnen, welche das Werk zensurieren sind pingelig. Für eine gute Abschlussnote wäre ein Gelingen sehr wichtig für mich, da die Gewichtung dieser Arbeit hoch ist.**

Antwort

Hier wären wir also schon fast bei der *Gretchen-Frage* angelangt. Gretchen-Fragen haben es so an sich, dass sie sich selten mit einem klaren Ja oder Nein beantworten lassen. Meistens braucht es dazu mehrere Teile der Tragödie. Nicht minder einfacher zu beantworten ist die Frage nach Alternativen zu MS Word. Eindeutig feststellen lässt sich: es gibt zu wenige Alternativen und diese Alternativen sind zu wenig verbreitet, was die Gefahr birgt, dass man sich in die Position des Einzelgängers begibt oder sogar mit seiner an sich sehr schönen Arbeit ins Abseits läuft.

Die Gefahr mit einem guten Werk, in das man «*nota bene*» viel Arbeit und Können gesteckt hat ins Abseits zu laufen, ist nicht zu unterschätzen. Niemand wird Ihnen um den Hals fallen, wenn Sie ein Werk abgeben, welches der Empfänger zuerst konvertieren muss, um so weniger, wenn dieser die Software zur Konvertierung zuerst beschaffen muss. Bedenken Sie, dass jede Konvertierung einen zusätzlichen Aufwand, ein bestimmtes Risiko (gerade bei grösseren und komplexeren Sachen) birgt und meistens auch einen Qualitätsverlust mit sich zieht. Diese Überlegungen sind allerdings zweitrangig, wenn von Ihnen ausschliesslich die Abgabe eines gedruckten Skriptes erwartet wird.

Seien und bleiben Sie deshalb kritisch gegenüber saloppen und pauschalen Aussprüchen, von Kollegen und Kolleginnen, die dann etwa lauten: "*MS Word kannste eh vergessen*" oder "*Machs doch einfach mit dem Pinguin-Programm".* Solche Sprüche hört man z.B. am Stammtisch. Es sind Aussagen aus dem Bauche heraus von Leuten, welche in diesem Bereich gar keine Verantwortung tragen bzw. einen etwaigen Fehlgriff nicht aussitzen müssen. Ein Programm, wie MS Word, welcher im Bereich PC-Textverarbeitung eine solch dominante Rolle einnimmt, kann man nicht einfach zur Seite schieben, viel mehr muss man sich hier der Realität stellen.

Es gibt übrigens tatsächlich Dinge, welche eine gute Dissertation eigentlich beinhalten sollte, mit MS Word aber nur sehr schlecht oder gar nicht zu realisieren sind. Ein guter, erfahrener und gerechter Zensor, sollte diese neuralgischen Punkte allerdings kennen und dürfte Ihnen daraus kein Strick ziehen.

Ohne Zweifel stellt MS Word ein sehr praktisches, beliebtes und auch gutes Werkzeug dar, um ein ein- bis zwanzigseitiges Dokument zu erstellen. Ist es aber auch das beste Programm um damit z.B. eine grosse Diplomarbeit mit komplexer Formatierung zu schaffen? Ist die *WYSIWYG (What You See Is What You Get)* Architektur, auf die sich MS Word stützt, und die zu einem guten Teil den wirtschaftlichen Erfolg dieses Produktes ausmacht, überhaupt dazu geeignet, wenn z.B. das ganze 200-seitige Dokument in einem sehr zeitraubenden Prozess jedes Mal neu umgebrochen wird, nur weil Sie die Schriftfarbe einer Formatvorlage ändern. Dieser Umbruch wäre zwar in diesem Falle absolut überflüssig (es wäre nur nötig, wenn die Schriftgrösse, die Zeichenweite oder die Abstände ändern), aber MS Word kann da offensichtlich nicht differenzieren.

Sie fügen ein kleines zusätzliches Bildchen ein. Word schaltet dazu in die Layoutansicht, aber erst nachdem die 200 Seiten neu umgebrochen wurden. Das ist eben auch *WYSIWYG*!. Eigentlich ein völliger Irrsinn, aber eben der Preis für die hohe Interaktivität solcher Anwendungen.

Der Preis scheint aber leider noch höher zu liegen. So ist es Microsoft in 15 Jahren nicht gelungen, dem Kunden eine stabile Word-Anwendung zu präsentieren. Auch beim jüngsten Kinde von Microsoft, *Word2000*, sind Programmabstürze im Zusammenhang mit grossen Dokumenten (fast) an der Tagesordnung. Die Gefahr der Dateikorruption ist gross und eine wohl organisierte Datensicherung durch den Benutzer ist schon deshalb unerlässlich.

Welches Textverarbeitungsprogramm für Sie das Beste ist, kommt sehr darauf an, welcher Typus von Benutzer Sie sind und das beste Programm ist oft das jenige, welches man am besten kennt.

Die folgende Tabelle, drückt eine persönliche Ansicht aus, könnte Ihnen aber bei der Entscheidung helfen.

Das geeignetste Textverarbeitungsprogramm für eine 150-seitige Dissertation mit komplexer Formatierung

Profil des Autors	Rang	Produkt	Prinzip	Bemerkungen
Der unsichere, unschlüssige Benutzer	**1.**	MS Word	WYSIWYG (*What You See Is What You get*)	Liegt immer noch im Trend. Sehr, sehr verbreitet Offensichtlich lassen sich lange Dokumente in MS Word erstellen, allerdings wird viel Schweiss vergossen und MS Word ist fast genau so lernintensiv, wie Anwendungen, welche nicht auf WYSIWYG beruhen. Was immer Sie für ein Produkt wählen, unterschätzen Sie die Zeit, welche Sie zur Erlernung des Tools brauchen nicht. Wenn Sie ungeübt sind, rechnen Sie damit, dass Sie ein zweites Mal, ja vielleicht sogar ein drittes Mal von vorne beginnen, weil Sie mit der Zeit realisieren, dass Sie vieles falsch angepackt haben.
	2.	---	---	Zögernden sollte man keine Alternativen auftischen
	3.	---	---	---
Der gut organisierte sehr ambitiöse Benutzer, welcher selten in Zeitnot gerät.	**1.**	Tex bzw. (LaText)	Kodierung und Plug-Ins	Die meisten wirklich professionellem Publikationen, werden mit dieser Software oder prinzipähnlichen Produkten realisiert. MS Word z.B. kann da nicht mithalten. Die Utilities müssen aber oft selbst zusammengesucht werden und kosten von gar nichts bis stattlicher Preis. Eignet sich sehr gut für technische und naturwissenschaftlicher Arbeiten.
	2.	HTML in Verbindung mit Acrobat Distiller von Adobe	Kodierung und dann Konvertierung	Mit HTML lernen Sie etwas, was Sie später sicher noch gebrauchen können. HTML ist super dokumentiert, steht jedem offen, korrupte HTML-Dokument gibt es nicht. Acrobat Distiller von Adobe schafft die Konvertierung zu einem publizierbaren Print-Medium fast perfekt. Ein riesiger Wermutstropfen ist allerdings, dass Formatierung, beruhend auf CSS (Cascading Sytle Sheets) -noch- nicht konvertiert werden kann. Ich hoffe aber und erwarte den CSS-Support bald. Technisch gesehen, sehe ich dazu nämlich keine speziellen Hürden, da es sich schlicht und einfach um ausgelagerte Kodierung handelt.
	3.	MS Word	WYSIWYG	Der Ambitiöse wird wohl bald sauer werden, wegen der vielen Fehlern in MS Word. So schafft es MS Word auch heute noch nicht in jedem Falle ein Inhaltsverzeichnis eines komplex strukturierten Dokumentes in jedem Fall korrekt zu erstellen. Auch mit Seitenzahlen und was damit zusammenhängt, steht MS Word auf dem Kriegsfuss. Das Definieren von Formatvorlagen ist enorm zeitaufwendig, unübersichtlich und wirkt auf versierte Benutzer altgebacken (geht über ein ganzer Dickicht von verästelten Dialogen). Formatvorlagen werden nicht wirklich geklont (pseudo-Lösung). Schlimmer noch ist die Instabilität sowohl der Software wie auch der Dokumente, welche mit dieser Anwendung erstellt werden. Dateikorruption ist häufig. WYSIWYG-Anwendungen können die Transparenz von Code natürlich nicht bringen. Man wundert und ärgert sich dann einfach, wenn etwas nicht so ist, wie es eigentlich sein sollte Eine vom korrupte Formatierungssequenz kann nur mit Hilfe von Probieren und Zufall aus dem Dokument eliminiert werden. Ein ungutes Gefühl bleibt dabei mit Recht zurück. Diese Problematik ist bei WYSIWYG-Anwendungen weitgehend systembedingt. Das Einbinden von Grafiken und Objekten ist zwar sehr flexibel gelöst, macht aber ein Dokument instabil. Der Zeitaufwand beim sauberen und gefälligen Einbinden von Grafiken ist hoch. Eignet sich eher schlecht für technische und naturwissenschaftliche Arbeiten. Der mitgelieferte Formeleditor hat grosse qualitative Mängel. Eingebettete Objekte dieser Art machen das Dokument instabil. Auch die Funktonalität des Formeleditors ist nicht überragend.

Profil des Autors	Rang	Produkt	Prinzip	Bemerkungen
Der auf Nummer sicher gehende Benutzer ohne spezielle Hingabe zur Erlernung eine Codier- oder Skript-Sprache	**1.**	MS Word	WYSIWYG	Weit verbreitet. Trotz aller Kritik hohe Akzeptanz. Schöne und auch gute Benutzeroberfläche. Schnelle Anfangserfolge.
	2.	StarOffice	WYSIWYG	Ist kaum schlechter als MS Word, allerdings schon viel weniger verbreitet. Die verschiedenen Anwendungen in StarOffice sind besser gegenseitig integriert, als in MS Office. Phänomenales Preis/Leistungsverhältnis, wobei man hinterfragen kann, wieso das Produkt (fast) verschenkt werden muss. Unsichere Zukunft. Klar positiver Trend zur weiteren Verbreitung seit Jahren nicht auszumachen.
	3.	WordPerfect	WYSIWYG	Hat zu recht seine Liebhaber, verliert aber immer mehr an Bedeutung Hat ein besseres Konzept bezüglich Tastenkürzel, womit Geübte flüssiger schreiben können und weniger zwischen Tastatur und Maus wechseln müssen. Das ist um so wichtiger, da es sich ja eigentlich um ein Textverarbeitungsprogramm handelt. Formatvorlagen, eines der wichtigsten Elemente in jedem Textverarbeitungssystem, können (auch) kodiert werden, und müssen nicht wie in MS Word über eine fast endlos lange Serie von Dialogen zusammen gestellt werden.
Meine persönlichen Favoriten	**1.**	HTML in Verbindung mit Acrobat Distiller von Adobe	Kodierung und dann Konvertierung	Endlich Licht in Sicht! Das Wischiwaschi WYSIWYG war mir immer sehr intransparent. Mir persönlich ist What You Define Is What You Get sympathischer. Zwar seit 15 Jahren ein wirtschaftlichen Erfolg, technisch aber eben so lange ein ziemlicher Flop.
	2.	MS Word	WYSIWYG	Ich kenne es halt gut und es ist sehr verbreitet. Bei keinem Vergleichsprodukt ist die Chance kompetente Hilfe zu bekommen so hoch. Für kleine Dokumente (1 - 20 Seiten) sehr gut geeignet. Hat eine Programmiersprache, welche sich stark an Visual Basic anlehnt. Obwohl im oberen Preissegment qualitativ ziemlich durchschnittlich. Feature-lastig zum Teil verspielt Mangelhafte bis sehr mangelhafte Stabilität. Ich sehe zu wenig Fortschritte bezüglich Qualität. Entwickelt sich weiter zur eierlegenden Wollmilchsau (Siehe Word2000: "Gestalten Sie Ihre Home Page mit MS Word").
	3.	Tex	Kodierung und Plug-Ins	Ich möchte Tex schon seit langem näher kennen lernen. Ähnelt stark den stabilen Textverarbeitungssystemen auf Grossrechnern, welche ich von früher gewohnt bin.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/LangDok.htm

René Probst, Mai 2000

Änderungen

- 7. Februar 2001 - Ergänzungen

Ein rotes Kreuz in einem Rechteck erscheint an der Stelle einer Grafik

◄ Zurück zum Hauptindex

Symptome

> Wo ich einst sorgfältig eine Grafik eingesetzt habe, erscheint nach dem erneuten Öffnen des Dokumentes nur noch ein Rechteck mit einem fetten roten Andreas-Kreuz.

Ursache

Word kann die gespeicherten Informationen zur Darstellung der Grafik nicht mehr interpretieren.

Abhilfe

- Besorgen Sie sich den Office Service Release 1 (SR-1) von Microsoft und installieren Sie diesen. Wie sie diesen Bugfix (von Microsoft beschönigend Service Release genannt) gratis beziehen und anschliessend installieren können, erfahren Sie weiter unten. In einigen Fällen werden die Grafiken nach erfolgreicher Installation des SR-1 wieder korrekt angezeigt. Der Ratschlag ist aber vor allem prophylaktisch gedacht. Der Service Release sorgt dafür, dass dieses Malheur in Zukunft weniger oft geschieht. Werden die Grafiken dann immer noch nicht richtig angezeigt, müssen Sie die Grafiken neu einbinden. Dieser Ratschlag hat sehr hohe Priorität, entfällt aber für Word2000.
- Manchmal hilft ein erneutes Öffnen des Dokumentes.
- Sorgen Sie für genügend Platz auf der Festplatte; und zwar sowohl auf dem Laufwerk auf dem Sie das Dokument abspeichern, wie auch auf dem Laufwerk auf dem das Betriebssystem (oft Windows) installiert ist. Das letztere ist wichtig, da die Auslagerungsdatei und temporäre Dateien bei einer typischen Installation, auf der primären Partition zu liegen kommen.
- Falls die Grafik verknüpft ist, markieren Sie den Platzhalter der Grafik und drücken Sie anschliessend die Taste **[F9]** ,um die Verknüpfung zu aktualisieren.
- Führen Sie einen Doppelklick auf den Platzhalter aus. Wenn sich der Grafikeditor nun öffnet und das Bild anzeigt, können Sie es über die Zwischenablage kopieren und erneut einfügen.
- Falls die Startdatei **autoexec.bat** den Befehl **SET TEMP = ...** beinhaltet, muss das dort gesetzte Verzeichnis auch wirklich existieren.

Vorbeugung

- Installieren Sie den Office Service Release 1 (hohe Priorität - entfällt aber bei Word2000)
- Sichern Sie Ihre Dokumente regelmässig (hohe Priorität)
- Verpassen Sie Ihren Dokumenten, welche einen gewissen, befriedigenden Stand erreicht haben einen Schreibschutz. Dies ist auch ein Selbstschutz, damit man nicht ständig, husch - husch, noch ein Detail ändert.
- Schalten Sie ganz generell die Schnellspeicherung unter ***Extras - Optionen - Speichern - Schnellspeicherung zulassen*** aus. Dieses Feature verursacht meiner Meinung nach mehr Problem als es hilft. Sie schalten damit die Autospeicherung nicht aus, allerdings dauert der Vorgang dann etwas länger. Das Ausschalten der Schnellspeicherung hat auch den positiven Effekt, dass sich Ihre Dokumente nicht mehr unnötig aufblasen.

Anmerkungen

- Dieses Problem kann übrigens jederzeit auch in **PowerPoint** auftreten.
- Hier handelt es sich um einer der ärgerlichsten Bugs in der Word- bzw. PowerPoint-Anwendung. Wenn eine Person öfters mit Grafiken in Worddokumenten zu tun hat, dann ist es in der Regel nur eine Frage der Zeit, bis sie Opfer dieser unschönen Sache wird. Ein einziger Absturz kann reichen, um das Problem hervorzurufen. Da Word nun wirklich nicht zu den stabilsten Anwendungen zählt, ist die Katastrophe eigentlich vorprogrammiert.
- Word hat es in dieser Sache auch nicht ganz einfach. Man bedenke, dass Word im Unterschied zu vielen Bildbetrachtungsprogrammen die Datei nicht nur zum Lesen öffnet, sondern als Datei mit Lese- und Schreibzugriff. Kommt hinzu, dass in einem Word-Dokument sehr viele Grafiken enthalten sein können, wogegen mit einem Grafikprogramm meistens nur ein einziges Bild gleichzeitig geöffnet wird. Im weiteren sind Dokumente, welche Grafiken enthalten meistens gross bis sehr gross, was die Anfälligkeit auf eine Dateikorruption weiter erhöht.
- Ob Word2000 in dieser Hinsicht besser mit Grafiken zurecht kommt als die Vorgängerversion, wird sich noch weisen müssen.
- Wirkliche Sicherheit bietet in solchen Situationen eigentlich nur **Redundanz**. Da ich nicht glaube, dass Word redundante Informationen speichert, um Grafiken wiederherstellen zu können, bleibt eigentlich nur die periodische Sicherung der Dateien übrig. Für diesen Vorgang müssen wir besorgt sein. Dieser Vorgang lässt sich allerdings weitgehend automatisieren.

Daten professionell Sichern

Mit dem Agenten für **geplante Vorgänge**, welcher z.B. in **Windows 98** enthalten ist, lässt sich die automatische und periodische Sicherung von Dateien doch recht gut automatisieren. Auf konzeptioneller Ebene sind dabei folgende Punkte zu beachten:

Idealerweise sollten die Dateien auf ein Medium gesichert werden, welches physisch vom Ursprungsort unabhängig ist (z.B. zweite Festplatte).

Die Praxis zeigt, dass mehrere Generationen von Sicherungssätzen vorhanden sein müssen, da oft eine gewisse Zeit verstreicht, bis das Problem entdeckt wird. Eine tägliche Sicherung, wobei die Datei unter dem selben Namen im gleichen Sicherungsverzeichnis abgelegt wird, erscheint im ersten Augenblick komfortabel, ist aber wertlos, sollte ich erst nach drei Tagen entdecke, dass die Ursprungsdatei korrupt ist. Der Frust könnte nicht grösser sein, denn diese Sicherungsdatei ist inzwischen genau so defekt, wie die ursprüngliche.

Schon besser eignet sich ein Konzept, welches auf dem Prinzip von verschieden Generationen beruht. Ich schlage also vor, folgende Sicherungssätze zu erstellen:

- Tägliche
- Wöchentliche
- Monatliche
- Halbjährliche

Dazu schreiben wir vier analoge Batch-Dateien, auch Stapeldateien genannt: Am besten geht dies mit dem kleinen Programm **notepad.exe**, welches sich im Lieferungsumfang von Windows befindet.

Jede Batch-Datei enthält zwei analoge Zeilen mit DOS-Befehlen. Die Pfadnamen müssen Sie selbstverständlich noch anpassen. Die Beispiele gehen davon aus, dass Sie Ihre Dokumente im Standardverzeichnis **C:\Eigene Dateien** und die Dokumentenvorlagen im Verzeichnis **C:\Office97\Office\Vorlagen** aufbewahren. Im weiteren wird vorausgesetzt, dass Sie über eine Festplatte verfügen, welche über eine Partition (Laufwerk) mit dem Laufwerkbuchstaben **G:** verfügt.

```
xcopy32 "C:\Eigene Dateien\*.*" "G:\Daily\Eigene Dateien\*.*" /s
xcopy32 "C:\Office97\Office\Vorlagen" "G:\Daily\Vorlagen\*.*" /s
```

(Diese zwei Zeilen speichern wir unter dem Namen SaveD und zwar im Stammverzeichnis.)

```
xcopy32 "C:\Eigene Dateien\*.*" "G:\Weekly\Eigene Dateien\*.*" /s
xcopy32 "C:\Office97\Office\Vorlagen" "G:\Weekly\Vorlagen\*.*" /s
```

(Diese zwei Zeilen speichern wir unter dem Namen SaveW und zwar im Stammverzeichnis.)

```
xcopy32 "C:\Eigene Dateien\*.*" "G:\Monthly\Eigene Dateien\*.*" /s
xcopy32 "C:\Office97\Office\Vorlagen" "G:\Monthly\Vorlagen\*.*" /s
```

(Diese zwei Zeilen speichern wir unter dem Namen *SaveM* und zwar im Stammverzeichnis.)

```
xcopy32 "C:\Eigene Dateien\*.*" "G:\Yearly\Eigene Dateien\*.*" /s
xcopy32 "C:\Office97\Office\Vorlagen" "G:\Yearly\Vorlagen\*.*" /s
```

(Diese zwei Zeilen speichern wir unter dem Namen *SaveY* und zwar im Stammverzeichnis.)

Notepad speichert diese Dateien mit der Dateierweiterung **.txt**. Dies ist aber für unser Vorhaben ungeeignet. Batch-Dateien müsste nämlich die Erweiterung **.bat** aufweisen. Wir benennen also die vier Dateien im Bezug auf die Dateierweiterung noch um. Dies gelingt uns am besten, wenn wir über Start **MS-DOS-Eingabeaufforderung** in den **DOS-Modus** wechseln. Im DOS-Modus wechseln wir in das Stammverzeichnis (auch root genannt) und geben nacheinander folgende vier DOS-Befehle ein (*ren* steht dabei für *rename*):

- `ren SaveD.txt SaveD.bat`
- `ren SaveW.txt SaveW.bat`
- `ren SaveM.txt SaveM.bat`
- `ren SaveY.txt SaveY.bat`

Sind alle vier Dateien erfolgreich umbenannt verlassen wir den DOS-Modus. Wir sind jetzt bereit, den Taskplaner zu benutzen. Als nächstes führen wir also einen Doppelklick auf den **Taskplaner** aus. Sie finden den Taskplaner üblicherweise als Symbol unten rechts am Bildschirmrand (dieser Bereich wird oft auch *Systray* genannt). Nun richten wir für jede der vier Batch-Dateien einen geplanten Vorgang ein.

Dabei gehen wir folgt vor:

1. Im Taskplaner führen wir einen Doppelklick auf den Eintrag **Geplante Task hinzufügen** aus damit startet der Assistent.
2. Wir betätigen die Schaltfläche ***Weiter***.
3. Im neuen Dialog betätigen wir die Schaltfläche ***Durchsuchen...*** und wechseln in das Stammverzeichnis.
4. In der Auflistung der Dateien suchen wir nun nach unserer Batch-Datei, also **SaveD** und öffnen diese mit einen Doppelklick.

5. Im nun folgenden Dialog wählen wir die Option ***Täglich*** und betätigen dann die Schaltfläche ***Weiter***.
6. Falls Bedarf besteht, können wir die Startzeit noch anpassen.
7. Wir schliessen nun den Vorgang ab, indem wir nacheinander die Schaltflächen ***Weiter*** und ***Fertig stellen*** betätigen. Damit haben wir die **tägliche Sicherung** eingerichtet.

1. Wir führen nun wiederum einen Doppelklick auf den Eintrag **Geplante Task hinzufügen** aus damit startet der Assistent.
2. Wir betätigen die Schaltfläche ***Weiter***.
3. Wir betätigen in diesem Dialog die Schaltfläche ***Durchsuchen...***.
4. In der Dateiauflistung suchen wir nun nach unserer Batch-Datei **SaveW** und führen einen Doppelklick auf diese aus.
5. Im nun folgenden Dialog wählen wir die Option ***Wöchentlich*** und dann die Schaltfläche ***Weiter***.
6. Wir wählen nun den Wochentag z. B. **Montag** aus. Falls Bedarf besteht, können wir die Startzeit noch anpassen.
7. Wir schliessen den Vorgang ab, indem wir nacheinander die Schaltflächen ***Weiter*** und ***Fertig stellen*** betätigen. Damit haben wir die **wöchentlich Sicherung** eingerichtet.

1. Wir führen erneut einen Doppelklick auf den Eintrag **Geplante Task hinzufügen** aus damit startet der Assistent.
2. Wir betätigen die Schaltfläche ***Weiter***.
3. Wir betätigen in diesem Dialog die Schaltfläche ***Durchsuchen...***.
4. In der Dateiauflistung suchen wir nun nach unserer Batch-Datei **SaveM** und führen einen Doppelklick auf diese aus.
5. Im nun folgenden Dialog wählen wir die Option ***Monatlich*** und dann die Schaltfläche ***Weiter***.
6. Wir wählen nun die Option **Am ersten Montag**. Falls Bedarf besteht, können wir die Startzeit noch anpassen.
7. Wir schliessen den Vorgang ab, indem wir nacheinander die Schaltflächen ***Weiter*** und ***Fertig stellen*** betätigen. Damit haben wir die **monatliche Sicherung** eingerichtet.

1. Wir führen abermals einen Doppelklick auf den Eintrag **Geplante Task hinzufügen** aus; damit startet der Assistent.
2. Wir betätigen die Schaltfläche ***Weiter***.
3. Wir betätigen in diesem Dialog die Schaltfläche ***Durchsuchen...***.
4. In der Dateiauflistung suchen wir nun nach unserer Batch-Datei, also **SaveY** und führen einen Doppelklick auf diese aus.
5. Im nun folgenden Dialog wählen wir die Option ***Monatlich*** und dann die Schaltfläche ***Weiter***.
6. Wir wählen nun die Option **Am ersten Montag**. Falls Bedarf besteht, können wir die Startzeit noch anpassen.
7. Wir deaktivieren nun die meisten Kästchen vor den zwölf Monatsnamen und lassen nur das Häkchen vor den Monaten **Jan** und **Jul** stehen.
8. Wir schliessen den Vorgang ab, indem wir nacheinander die Schaltflächen ***Weiter*** und ***Fertig stellen*** betätigen. Damit haben wir die **halbjährliche Sicherung** eingerichtet.

Nachdem wir unsere Arbeit sorgfältig geprüft haben, können wir den Taskplaner verlassen. Die Überprüfung geht übrigens am einfachsten in der Detail-Ansicht des Taskplaners. Wir können aber auch noch etwas im Taskplaner verweilen und alle vier Vorgänge als ersten Test, forciert auslösen. Dies erreichen wir, wenn wir auf eine der Tasks einen Rechtsklick ausüben und im nun erscheinenden Kontextmenü **Starten** wählen.

Damit diese Vorgänge uns bei der Arbeit nicht stören, veranlassen wir zum Schluss noch, dass die *Tasks* im Hintergrund ablaufen. Dazu gehen wir wie folgt vor:

1. Im Windows-Explorer suchen nach unseren Batch-Dateien, welche seit ihrer Umbenennung unter einem neuen Ikon (Sinnbild) sichtbar sind und führen folgende Nacharbeiten durch:
2. Wir führen einen ***Rechtsklick*** auf die Batch-Datei aus, welche wir erstellt haben. Im nun erscheinenden Kontextmenü wählen wir ***Eigenschaften***.
3. Wir wählen den Reiter ***Programm***.
4. Unter ***Ausführen*** wechseln wir den Eintrag von *Normales Fenster* auf **Minimiert**.
5. Zusätzlich aktivieren wir das Kästchen **Beim beenden schliessen**.
6. Mit ***Ok*** verlassen wir diesen Dialog wieder.

Diese Aktion sollte für alle vier Batch-Dateien durchgeführt werden.

Beachten Sie, das eine bestimmte Sicherung nur stattfindet, falls der Rechner zum Zeitpunkt den wir im Task-Manager eingegeben haben, eingeschaltet ist. Deshalb sollte die Ausführungszeit der Tasks klug gewählt werden.

Wenn wir nun auf eine Wiederherstellung von einer oder mehreren Dateien zurückgreifen müssen, begutachten wir zuerst die entspreche Sicherung im Verzeichnis "Daily", da es sich dabei um die jüngste Sicherung handelt. Falls der Zustand der fraglichen Datei dort nicht zufriedenstellend ist kommt als nächstes der Ordner "*Weekly*" an der Reihe. In diesem Sinne fahren wir fort, bis wir eine intakte Sicherungsdatei aufgefunden haben.

Wie viel Platz Sie auf der Festplatte für alle Sicherungssätze opfern müssen, ist natürlich sehr individuell. 300MB könnten aber einen guten Start abgeben. Sollten Sie sich entschlossen habe, alle vier Generationen für den Beginn von Hand auszulösen, wissen Sie natürlich besser Bescheid. Seien Sie grosszügig und gehen Sie davon aus, dass der Datenbestand mit der Zeit noch wachsen wird. Genügend Platz ist natürlich wichtig für ein Gelingen der Sicherung.

Links zu diesem Thema

Microsoft

Deutsch

► **Wd97: Grafiken werden im Word Dokument durch rote Kreuze ersetzt**

Englisch

► **WD97: Picture Displayed as Red "X" in Document**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/RotesX.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Der Fussnotentext wird nicht auf der richtigen Seite gedruckt

◄ Zurück zum Hauptindex

Symptome

> Der Fussnotentext wird nicht auf die richtige Seite gesetzt, sondern um eine Seite nach unten geschoben. In der Regel erscheint die dazugehörige Zahl zwar noch auf der richtigen Seite, der Text aber auf der nächsten. Damit wirkt das Dokument wie zerschossen.

Ursache

Regen Sie sich nicht unnötig auf, diese Erscheinung ist in Word schon fast die Norm. Dieser Bug spuckt schon seit den Zeiten von MS Word für DOS 2.0 in dieser Anwendung herum.

Abhilfe

- Richten Sie die Formatvorlage für den **Fußnotentext** und die Formatvorlage für den **Haupttext** neu ein. Dabei ist folgendes zu beachten:
    1. Stellen Sie unter **Format - Formatvorlage - Bearbeiten - Format - Absatz - Abstand** den Zeilenabstand auf **Genau**. Damit es ordentlich aussieht, also ein passender Durchschuss (Zeilenabstand) entsteht, können Sie nach belieben das Mass für den Zeilenabstand so setzen, dass dieser der Schriftgrösse plus ein- oder zwei Punkte entspricht. Ist die Schriftgrösse also 10pt. Stellen Sie unter Maß 11 oder 12pt. ein.
    2. Stellen Sie unter ***Format - Formatvorlage - Bearbeiten - Format – Absatz*** ale aufweisen: (Sie finden all diese Einstellungen unter Absatz - Zeilen- und Seitenwechsel).
        - Absatzkontrolle an
        - Zeilen nicht trennen an
        - Absätze nicht trennen an

Anmerkung

In den meisten Fällen helfen die hier beschriebenen Einstellungen weiter, das heisst die Platzierung des Fussnotentextes stimm dann. Zur Sicherheit sollten Sie abklären, ob dies auch der Fall ist, wenn das Dokument auf einem anderen Drucker ausdruckt wird.

Links zu diesem Thema

Microsoft

Englisch

► WD: Footnote Moved to Next Page

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/FNBug.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Es stehen keine oder nur zwei Schriften zur Verfügung

◄ Zurück zum Hauptindex

Symptome

> **Obwohl Sie eine stattliche Zahl von Schriften im Schriftenordner installiert haben, stehen Ihnen im Schriftenkombinationsfeld auf der Word-Symbolleiste Format nur zwei zu Verfügung, namentlich: *Times New Roman* und *Roman 10cpi*.**

Ursache

Sie haben unter Windows keinen oder keinen grafikfähigen Drucker installiert (also z.B. den Drucker *Universal/Nur Text*).

Abhilfe

Installieren Sie unter Windows einen **grafikfähigen** Drucker und zwar auch dann, wenn Sie auf diesem Rechner gar keinen Drucker betreiben (z.B. bei einem Laptop). Zusätzlich sollten Sie diesen Drucker zum **Standarddrucker** küren.

Der Druckertreiber sollte Ihrem angeschlossenen Drucker möglichst genau entsprechen. Falls kein Drucker angeschlossen ist, wählen Sie z.B. einen der *HP DeskJet Drucker*, welcher in der Treiberauswahlliste von Windows enthalten ist.

So installieren Sie einem Drucker unter Windows (98)

Falls der Hersteller des Druckers kein spezielles Vorgehen zur Installation des Druckers vorsieht, gehen Sie wie folgt vor:

1. Führen Sie einen Doppelklick auf das Symbol ***Arbeitsplatz*** aus.
2. Führen Sie nun einen Doppelklick auf den Systemordner ***Drucker*** aus.
3. Führen Sie nun einen Doppelklick auf das Symbol ***Neuer Drucker*** aus.
4. Unter Windows 98 startet jetzt ein Assistent. Fahren Sie mit ***Weiter*** fort.
5. Wenn Sie nicht auf einen Drucker im Netzwerk drucken wollen, verwenden Sie die Option **Lokaler Drucker** und fahren Sie mit ***Weiter*** fort.
6. Markieren Sie in der linken Liste zuerst den **Hersteller** des Druckers und in der rechten Liste das **Modell** Ihres Druckers und fahren Sie mit ***Weiter*** fort.
7. Wählen Sie im nächsten Dialog den Anschluss (es handelt sich dabei um die Schnittstelle an dem der Drucker physisch angeschlossen ist). Sind Sie sich in diesem Punkt nicht schlüssig wählen Sie hier **LPT1:** und fahren Sie mit ***Weiter*** fort.
8. Übernehmen Sie im nächsten Dialog den Namen des Druckers, wie er Ihnen vorgeschlagen wird und wählen Sie die Option *Als Standarddrucker einrichten* (sinngemäss)
9. Fahren Sie zuerst mit ***Weiter*** und anschliessend mit ***Fertig stellen*** fort. Dabei werden Sie vielleicht aufgefordert, die Windows-CD in das Laufwerk einzulegen. Kommen Sie dieser Aufforderung nach.

So küren Sie unter Windows (98) einen Standarddrucker

1. Führen Sie einen Doppelklick auf das Symbol Arbeitsplatz aus.
2. Führen Sie nun einen Doppelklick auf den Systemordner Drucker aus.
3. Führen Sie einen Rechtklick auf den Drucker in der Liste aus, welcher zum Standarddrucker werden soll.
4. Wählen Sie im nun erscheinenden Kontextmenü den Punkt: Als Standard definieren.

Anmerkung

- Word 97 und Word2000 arbeiten sehr eng mit dem jeweiligen Druckertreiber zusammen, deshalb ist die Installation eines Druckers unumgänglich. Das hier beschriebene Phänomen ist übrigens auch in WordPad zu beobachten, nicht aber in Excel und in anderen Anwendungen, welche offensichtlich nach einer anderen Methode auf den Schriftenordner zugreifen.

Links zu diese, Thema

Microsoft

Englisch

► **Missing TrueType Fonts in *WordPad* and Microsoft Word**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/KeineSch.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Die Seitenränder liegen ausserhalb des bedruckbaren Bereiches! Wieso kann ich nicht bis zum Blattrand drucken?

◄ Zurück zum Hauptindex

Problem

> **1. Wenn ich mein Dokument ausdrucke erscheint eine Warnung, welche wie folgt lautet:**
>
> ***Die Seitenränder liegen im Abschnitt x ausserhalb des bedruckbaren Bereiches. Soll der Vorgang fortgesetzt werden?***
>
> **Gedruckt wird dann doch, wenn ich bestätige. Aber das häufige Auftreten dieser Warnung schikaniert mich.**
>
> **2. Ich möchte die Seitennummern in einem schattierten Rechteck darstellen. Dabei soll diese Schattierung bis ganz an den Blattrand reichen. In der Seitenlayoutvorschau gelingt mir das auch, trotzdem bleibt beim Druck ein weisser Rand von markanter Breite.**

Antwort

Es gibt wohl keinen Drucker, welcher das Blatt Papier ganz ausnutzen könnte. Wenn Sie diese Art von Layout z.B. in einer Broschüre beobachten, dann ist dass so zu Stande gekommen, dass mehrere Seiten auf einen grossen Bogen Papier gedruckt wurden. Anschliessend wurde dieser Bogen Papier mit einer Präzisionsmaschine in die einzelnen Seiten aufgeschnitten

Die Warnung, dass die Seitenränder ausserhalb des bedruckbaren Bereich liegen, ist gerechtfertigt, wenn Sie die Seitenränder zu knapp einstellen. Dazu sollte man wissen, dass MS Word sehr eng mit dem jeweiligen aktiven Druckertreiber zusammen arbeitet. Die Alternative dazu - also keine Warnung - würde wohl in der Praxis zu unnötigem Papierabfall führen.

Erklärungen

Man kann die Architektur dieses Systems in folgende drei Ebenen aufteilen:

- Die unterste Ebene ist der Drucker selbst, bzw. dessen Mechanik. Das Blatt muss während des Druckvorganges festgehalten und - sollte es sich nicht um einen Laserdrucker handeln - auch transportiert werden. In den äussersten Bereichen des Blattes ist somit kein Bedrucken möglich. Laserdrucker schneiden besser ab, was den maximalen bedruckbaren Bereich betrifft, als z.B. Tintenstrahldrucker, welche das Papier beim Druckvorgang laufend transportieren müssen.

- Die mittlere Ebene ist der Druckertreiber, also die Software, welche bei der Installation des Druckers eingespielt wurde. In diesem Treiber ist festgehalten, wie breit die Seitenränder im einzelnen mindestens sein sollten. Der Hersteller hat sich vermutlich Mühe gegeben, den Druckertreiber möglichst perfekt auf die Hardware (also auf den Druckers) abzustimmen.

- Die höchste Ebene ist die Anwendungssoftware, in unserem Falle also MS Word. Die Entwickler von Word haben sich entschlossen, die Informationen direkt und dynamisch aus dem Druckertreiber zu ziehen. Das will heissen, dass MS Word sehr eng mit dem jeweilig aktiven Druckertreiber zusammen arbeitet. Diese Ebene ist damit, was MS Word betrifft, nicht von grosser Bedeutung.

Lösung

Warnung vermeiden

Um die Warnung zu vermeiden, gehen Sie im Einzelnen wie folgt vor:

1. Rufen Sie vom Word-Hauptmenü den Dialog ***Datei - Drucken...*** auf und aktivieren Sie den fraglichen Drucker.
2. Rufen Sie nun den Dialog Datei - Seite einrichten... - Seiteränder auf.
3. Geben Sie nun für alle vier Seitenränder den unrealistischen Wert **0** ein. Word wird nun diese Einstellung monieren. Drücken Sie auf der erscheinenden Meldungsbox die Schaltfläche mit der Bezeichnung **Korrigieren**. Damit trägt Word die kleinstmöglichsten Werte selbständig ein. Nochmals: Word bezieht diese Informationen aus dem Druckertreiber.

☛ Wenn Sie im besagten Dialog zusätzlich die Schaltfläche mit der Bezeichnung **Standard...** klicken, dann gilt diese Einstellung fortan für alle Dokumente, welche auf Basis der Standardvorlage mit dem Namen **Normal.dot** erstellt werden.

Anmerkungen

- Beachten Sie, dass diese Einstellungen drucker- bzw. druckertreiberspezifisch sind. Allenfalls sollten Sie das soeben beschriebene Vorgehen für andere Drucker wiederholen.
- Greifen Sie allenfalls auf die (technische) Bedienungsanleitung des Druckers zurück. Auch dort sollten diese Parameters dokumentiert sein.
- Bei Tintenstrahldruckern ist beim hochformatigem Druck meistens der untere Rand besonders kritisch. So erwarten z.B. viele *HP DeskJet* Drucker einen unteren Rand von stattlichen **1.79 cm**.
- Beim Druck im Querformat sollten Sie umdenken. Dort ist der linke oder rechte Rand kritisch, je nach dem, in welche Richtung der Druckertreiber das Schriftbild um 90° dreht.
- Achten Sie auch darauf, dass die **Entfernung der Fusszeile** zum Blattrand genügend gross ist, um nicht zu riskieren, dass etwas abgeschnitten wird. Schauen Sie in diesem Zusammenhang auch die Seitennummern an, falls diese links- oder rechtsbündig sind. Überprüfen Sie dies nicht nur auf der ersten Seite des Ausdrucks, sondern auch auf der hundertsten, wo die Seitennummer mehr Platz beanspruchen.

Bis an den Rand drucken

Die Vorgabe, bis an den Blattrand eine Schattierung zu drucken, wie dies in der ursprünglichen Fragestellung suggeriert wird, ist unrealistisch. Falls Sie an diesem Vorhaben fest halten möchten, vergeben Sie diesen Auftrag an eine gut ausgerüstete Druckerei.

Warnung beim Druck ex VBA unterdrücken

Während diese Warnung bei einem manuellen Vorgang nur lästig ist, führt sie bei einem automatisierten Prozess dazu, dass der Vorgang unvermittelt abbricht, bzw. für unbestimmte Zeit auf die Eingabe einer Person wartet.

Um zu vermeiden, dass dies mit Ihrer VBA-Routine geschieht, müssen Sie zwei Vorkehrungen treffen. Einerseits müssen Sie sicher stellen, dass Word nicht geschwätzig ist, und anderseits müssen Sie dafür sorgen, dass der Druck **synchron**, also **nicht im Hintergrund** geschieht, da das Unterdrücken von Warnungen nicht an einen asynchronen Prozess weiter delegiert wird. Vergessen Sie nicht, die Standardwerte des Benutzers wieder zurück zu stellen, wenn Ihre Routine ihren Job gemacht hat, am Besten gleich unmittelbar nach dem Ausdruck. Die eben erwähnten Vorkehrungen reflektiert der folgende VBA-Code.

```
Sub DruckenUndWarnungIgnorieren()
  savEnvAlert = Application.DisplayAlerts
  Application.DisplayAlerts = wdAlertsNone
  ActiveDocument.PrintOut Background:=False
  Application.DisplayAlerts = savEnvAlert
End Sub
```

(Damit ein automatisierter Vorgang nicht unterbrochen wird, muss vor dem Ausdruck dafür gesorgt werden, dass Warnungen von Word unterdrückt werden. Zusätzlich muss aber auch sicher gestellt werden, dass im Vordergrund gedruckt wird.)

Links zu diesem Thema

Microsoft

Deutsch

► **WD2000: Seitenränder außerhalb bedruckbaren Seitenbereichs**

Englisch

► **WD2000: ErrMsg: "The margins of section n are set outside the printable area of the page."**

► **WD97: DisplayAlerts=wdAlertsNone Fails to Suppress Print Error**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SR.htm

René, Probst, Februar 2001

Änderungen

- --- nil ---

Wie kann ich aus einem Word-Dokument ein Büchlein erzeugen?

◄ Zurück zum Hauptindex

Problem

Ich möchte aus einem 16-seitigen Word-Dokument ein Büchlein im A5-Format erzeugen. Dafür möchte ich auf A4 Papier drucken, die Blätter dann falten und heften. Nun erlaubt mir zwar Word2000 den Druck von 2 Seiten pro Blatt, allerdings ist dann die Reihenfolge der Seiten nicht passend. Dies in Eingedenk, dass die Blätter gefaltet und ineinander gelegt werden müssen.

So sieht das Ergebnis aus, wenn Sie unter Word2000 den Befehl *Drucken – Zoom – 2 Seiten pro Blatt* wählen. Damit ist aber das Erstellen eines Büchleins nicht möglich...

Antwort

Mit **Word2000** ist es nun tatsächlich möglich, zwei (oder auch mehr) Seiten auf ein Blatt zu drucken. Leider löst dies die Eingangs geschilderte Problematik aber nicht. Benutzer von Word2000 sind zwar gegenüber Anwender von Word 97 hier etwas besser bedient, es reicht allerdings nicht aus, um ein *Booklet* oder eine Broschüre zu erstellen. Wenn Sie keine Druckerei kennen, welche Ihnen dies zu einem fairen Preis erledigen kann, dann sind Sie nach wie vor auf zusätzliche Software angewiesen.

Die beiden Abbildungen (links und rechts), sollen die Problematik, welche sich beim Druck <BOOKLETS< i>stellt, nochmals verdeutlichen.

Alternativen

Ich kenne drei kostenmässig vertretbare Möglichkeiten, das Ziel mit zusätzlicher Software zu erreichen.

MS Publischer2000

MS Publisher ist eine Anwendung, welche Sie einzeln im Handel erwerben können. Die *MS Office2000 Premium Edition* beinhaltet dieses Programm bereits. Der Publisher verfügt über einen Assistenten, welcher die Erzeugung von *Booklets* vorsieht. Den Text, welchen Sie vermutlich bereits geschrieben haben, müssten Sie dann von MS Word nach MS Publisher übernehmen. Beachten Sie dass die Operfläche und das Verhalten des Publisher sich doch recht stark von MS Word unterscheidet. Erfahrungsgemäss erfordert die Übernahme der Texte nach MS Publisher recht viel Zeit, insbesondere dann, wenn dies möglichst layoutgetreu geschehen soll. Deshalb steht diese Möglichkeit für mich nicht im Vordergrund.

...es müsste viel mehr etwa so aussehen. Dies ist aber mit Word2000 nicht machbar.

Übrigens unterstützen auch schon ältere Versionen des MS Publishers (Publisher 98) den Druck von *Booklets*.

Shareware *FinePrint 2000*

FinePrint ist eine sehr populäre und erprobte *Shareware*. Sie können eine eingeschränkte Testversion vom Netzt herunterladen und dann auch gleich testen. Möchten Sie diese Software aber über längere Zeit und ohne funktionellen und optischen Einschränkungen nutzen, müssten Sie sich um eine Volllizenz kümmern, welche ca. mit € 45,00 zu Buche schlägt. Die Download-Adresse dieses Produktes, finden Sie ganz am Schluss dieses Skriptes.

FinePrint und das Produkt, welches im nächsten Kapitel vorgestellt wird, basiert übrigens auf einem sehr raffinierten Konzept. Die Software mischt sich nämlich nicht etwa auf Basis von MS Word oder anderen Anwendungen in die Sache ein, sondern stellt einen Druckertreiber dar. Statt den Ausdruck direkt an Ihren Drucker zu leiten, wird der Ausdruck also zuerst an *FinePrint* geschickt.

Nach der Installation der Software, werden Sie also im Dialog ***Datei - Drucken*** einen neuen, virtuellen (gedachten) Drucker vorfinden. Für meine Begriffe, stellt dies die einzige wirklich gute Methode dar, diese Aufgabenstellung zu lösen. Das Konzept hat vor allem den Vorteil, dass die Funktionalität in **jeder Windows-Anwendung**, welche den Ausdruck ermöglicht, zur Verfügung steht. Im weiteren handelt es sich um ein Lösungsansatz, welcher von einem allfälligen Versionenwechsel (z.B. von MS Word) kaum tangiert sein wird.

FinePrint ist sehr einfach zu installieren und zu nutzen. Wenn Sie nach der Installation das erste mal ein *Booklet* erstellen, kommt ein Assistent zum Zuge, welcher alle die zum Druck eines Büchleins notwendigen Schritte durchgeht. Dieser Einrichtungsvorgang wäre ohne Assistenten gar nicht so einfach zu bewerkstelligen, unterscheiden sich doch die auf dem Markt erhältlichen Drucker sehr, was die mechanische und elektronische Steuerung betrifft. Es gibt Duplex-Drucker, also Drucker, welche das Blatt beidseitig bedrucken können, in dem sie das Papier automatisch wenden. Diese Drucker müssen bei der Aufgabenstellung anders gesteuert werden, als Simplex-Drucker. Es gibt aber in diesem Bereich weitere wichtige Unterschiede, insbesondere ist es auch ausschlaggebend, in welcher Richtung das Schriftbild , beim Querformatdruck rotiert werden muss. Der bereits erwähnte Assistent, spielt mit Ihnen, alle diese Faktoren, welche für ein Gelingen wichtig sind, durch und legt das Resultat dieser Tests auf Ihrem Rechner ab, um in Zukunft darauf zurück greifen zu können.

FinePrint kann nicht nur *Booklets* erzeugen, sondern kennet noch andere Tricks und Kniffe wie die folgende Liste aufzeigen soll.

Funktionelle Beschreibung (Auszug)

- Bis zu acht Seiten pro Blatt drucken
- *Booklets* drucken
- Seitenrahmen drucken
- Das Papier mit einem Wasserzeichen versehen
- Das Papier mit einer Grafik versehen, welche sich auf jeder Seite wiederholt

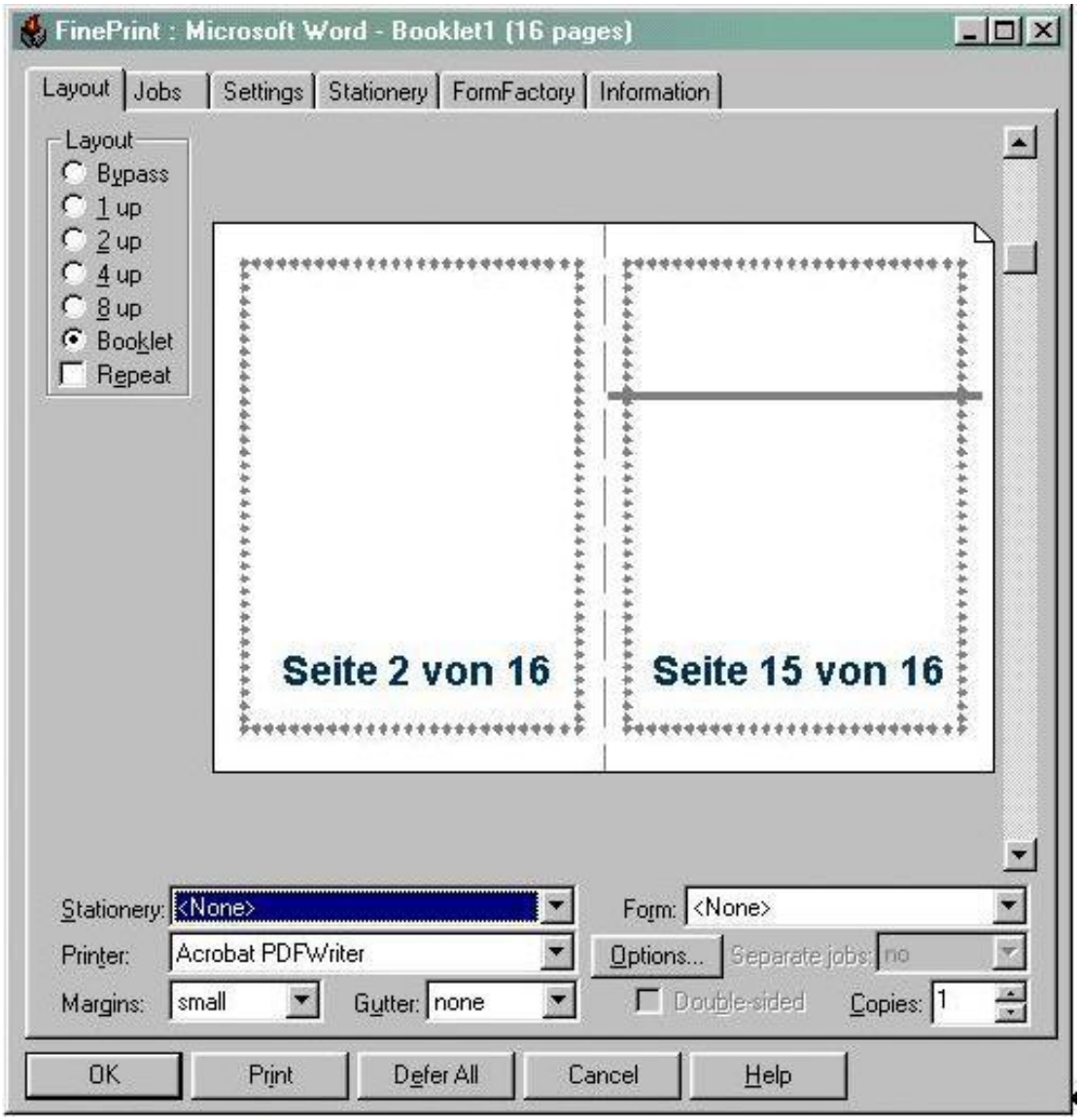
Wie schon erwähnt, nach der Installation der Software werden Sie, wenn Sie z.B. in MS Word den Dialog ***Datei - Drucken*** wählen feststellen, dass ein zusätzlicher Drucker zur Verfügung steht, welcher sich ***FinePrint 2000*** nennt.

Anmerkungen

- Einige der erwähnten Funktionen sind in MS Word auch ohne zusätzliche Software zu erreichen. Dank Programmen wie *FinePrint* funktioniert dies dann aber auch aus anderen beliebigen Anwendungen.
- Um mit *FinePrint* ein *Booklet* drucken zu können, Bedarf es der neusten z.Z. erhältlichen Version nämlich **FinePrint 2000**. Frühere Versionen können also noch keine Büchleins erzeugen.

Benutzeroberfläche

Die folgende Abbildung zeigt die Benutzeroberfläche der Software *FinePrint* 2000



Shareware *ClickBook 2000* bzw. *ClickBook 2001*

Etwas weniger verbreitet, aber bei den Benutzer sehr beliebt, ist die *Shareware* mit dem Namen **ClickBook**, welche vom Hersteller ***BlueSquirel*** stammt. Der Preis für das Produkt ist etwas höher als bei *FinePrint* nämlich ca. € 55,00. Der Funktionsumfang ist allerdings auch viel grösser. Die Benutzeroberfläche ist näher an jener, welcher wir von typischen Windows-Anwendungen gewohnt sind und welche wir schätzen gelernt haben. Insbesondere ist die Auswahl von möglichen *Layouts*, unter welchen Sie auswählen können, viel grösser und lässt eigentlich kaum Wünsche offen.

Die Arbeitsweise ist die selbe, wie jene, welche hier bereits beim Produkt *FinePrint* geschildert wurde. Auch hier erleichtert ein Assistent das Einrichten. Damit ist gewährleistet, dass die Software mit den Gegebenheiten Ihres Drucker zu Recht kommt und dass Ihnen sehr schnell Resultate gelingen.

Die Download-Adresse für diese Produkt, finden Sie am Ende dieses Skripts. Sie können die Lösung fünf Tage lang ohne Einschränkungen testen. Wollen Sie das Produkt dann weiter verwenden, müssen Sie eine Lizenz zum bereits erwähnten Preis erwerben.

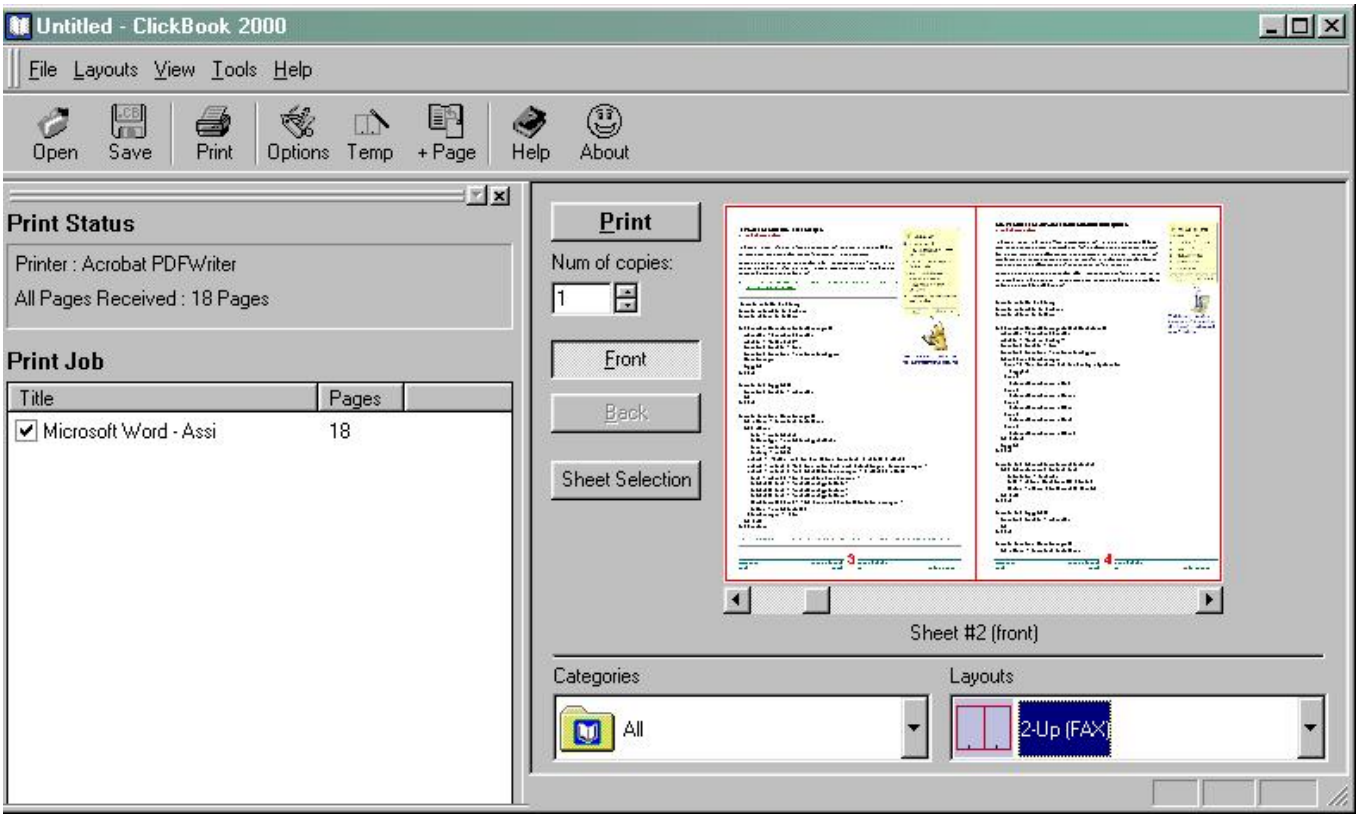
Funktionelle Beschreibung (Auszug)

In der folgenden Liste finden Sie einen kleinen Überblick, was diese Software leistet:

- Bis zu acht Seiten pro Blatt drucken
- *Booklets* drucken
- Seitenrahmen drucken
- Faltkarten drucken
- Visitenkarten drucken
- Umschläge drucken
- CD-Einlagen drucken

Benutzeroberfläche

Die folgende Abbildung zeigt die Benutzeroberfläche der Software ClickBook 2000:



Nach der Installation der Software werden Sie, wenn Sie z.B. in MS Word den Dialog ***Datei - Drucken*** wählen feststellen, dass ein zusätzlicher Drucker zur Verfügung steht, welcher sich *ClickBook*. nennt.

Zusätzlich werden Sie unter MS Word und Excel eine neue Schaltfläche (Symbol) vorfinden, welche es ebenfalls ermöglichen soll, auf diesen Druckertreiber zuzugreifen. Diese Schaltfläche hat allerdings bei meinen Versuchen (unter Windows 98 und MS Office2000) nicht zum gewünschten Erfolg geführt. Vielmehr resultierte beim Klick auf diese neue Schaltfläche eine Fehlermeldung im Sinne: *Konnte den aktiven Drucker nicht ändern*. Das Drucken auf die neue Schnittstelle über den Umweg *Datei - Drucken....* funktionierte dagegen tadellos. Ich möchte erwähnen, dass diese zusätzliche Schaltfläche eigentlich gar nicht gebraucht wird, um die Software zu nutzen. Ich habe dann auch die Dokumentvorlage mit dem Namen **cb97.dot**, welche diese Schaltfläche unterstützen sollte, wieder aus dem Office *StartUp*-Ordner entfernt.

Achten Sie beim Herunterladen von *ClickBook*, dass Sie die Version erwischen, welche für Ihr Betriebssystem geschaffen ist. Für Windows2000 scheint es bereits die Version *ClickBook 2001* zu geben, für ältere Windowsversionen heisst das Produkt z.Z. noch *ClickBook 2000*.

Persönlich Empfehlung

Beide Produkte erfüllen die Erwartungen und sind ihr Preis sicher auch Wert. Ausser im Preis hat ***ClickBook*** in allen Kategorien die Nase vorne, namentlich was Funktionalität, Benutzeroberfläche und Qualität des Ausdruckes betrifft. In meinem Test hat es sich gezeigt, dass *ClickBook* eine besser Auflösung zurück lässt. Entsprechend ist dann auch die Datenmenge grösser, welche an den Drucker-Spool geschickt wird.

Meine Empfehlung lautet als klar für *ClickBook*. Der eher kleine Preisunterschied von ca. 15% ist gerechtfertigt, weil diese Software auch mehr zu bieten hat, als das Konkurrenzprodukt. Für *FinePrint* allerdings spricht, dass die Software sich sehr, sehr einfach bedienen lässt.

Wieso keine Makrolösung mit Word-VBA?

Steht noch die Frage im Raum, wieso es sinnvoll bzw. nötig ist, eine zusätzliche Software einzusetzen, verfügt doch MS Word über eine Schnittstelle, welche es theoretisch ermöglichen würde, diese Angelegenheit über eine VBA-Routine zu lösen. Überlegt man sich die Sache allerdings genauer und auch konkreter, kommt man zum Schluss, dass sehr vieles gegen eine Makrolösung spricht. Die folgende Auflistung soll dies verdeutlichen:

- Der Lösungsansatz, die Skalierung auf Ebene Druckertreiber zu vollführen ist wesentlich sinnvoller, als dies auf der Ebene der Word-Objekte zu tun. Damit entfallen zahlreiche Abhängigkeiten. Dieser Weg funktioniert unabhängig von den zahlreichen Fehlern und Widerwertigkeiten, welche MS Word und Word-VBA im negativen Sinne leider auszeichnen.
- Word-Dokumente können eine Vielzahl von Objekten enthalten wie z.B. Tabellen, Grafiken u.a.m.. Alle diese Objekte müssten skaliert werden. Diese Umsetzung erfordert sehr grosse Kenntnisse des Word-Objekt-Modells und würde sehr viel Programmierarbeit abfordern. Ein solches Werk ist kaum bezahlbar, es sei denn man zieht das Projekt kommerziell auf. In diesem Falle kommt man aber bei der Analyse schnell zum Schluss, dass ein Makro der falsche Weg ist, um die Sache anzugehen.
- Einfacher und billiger wäre es selbstverständlich, sich auf eine auserwählte Anzahl von Objekten zu beschränken. Es ist zweifelsohne so, dass kaum ein Word-Dokument alle potentiellen Gestaltungsmittel einsetzt. Nur, auf welche Objekte soll man sich beschränke? - Diese Frage beantwortet wohl jeder mögliche Anwender anders. Die Gefahr, dass die Anforderungen an ein solches Makro sukzessive steigen und dass ein solches Werk als Unvollendete in die Geschichte eingeht, ist sehr hoch.
- Eine Makrolösung wäre nicht zukunftssicher kommen doch mit jeder neuen Word-Version neuartige Elemente hinzu.
- Eine Makrolösung müsste sehr intensiv getestet werden. Dies müsste mit vielen Dokumenten geschehen, auch Dokumente, welche von unterschiedlichen Personen mit unterschiedlichen Gewohnheiten erzeugt wurden. Dies stellt eine Heidenarbeit dar. Stellen Sie sich mal vor, Sie müssten duzende von Dokumente, deren Inhalt Sie vielleicht gar nicht motiviert, penibel auf Fehler was die Gestaltung betrifft, durchsuchen.
- Eine Makrolösung wäre fehleranfällig und langsam in der Ausführung.
- Die Word-Makro-Lösung stünde ausschliesslich in MS Word zur Verfügung. Andere Anwendungen könnten davon nicht profitieren.

Links zu diesem Thema

FinePrint

► **FinePrint Home Page**

ClickBook

► **ClickBook Home Page**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Booklet.htm

René, Probst, März 2001

Änderungen

- --- nil ---

In der Kopf- oder Fusszeile erscheint die unsinnige Angabe Seite 7 von 1

◄ Zurück zum Hauptindex

Symptome

> In der Seiten-, Seitenlayout- oder Kopf-/Fusszeilen-Ansicht erscheinen unsinnige Gesamtseitenzahlen wie z.B. *Seite 7 von 1*. Wird das Dokument gedruckt stimmen diese Zahlen aber.

Ursache

Word kann die Gesamtseitenzahl nur ermitteln, nachdem es für das ganze Dokument den Seitenumbruchvorgang durchgeführt hat. Da dieser Vorgang aber bei komplexeren Dokumente viel Zeit in Anspruch nimmt, während dessen an einer Weiterbearbeitung am Dokument nicht zu denken ist, verzichtet Microsoft aus Gründen der Performanz darauf, zum Beispiel nach dem Öffnen eines Dokumentes, diesen Vorgang zu initialisieren.

Abhilfe

- Wechseln Sie in die **Seitenansicht** und beenden Sie diese Ansicht anschliessend mit der Schaltfläche ***Schliessen***. Unter Umständen müssen Sie diesen Vorgang zwei oder drei Mal wiederholen, bis die Seitenzahlen zum Stimmen kommen. Bei einer typischen Word-Installation finden sie den Befehl Seitenansicht, gut erreichbar, auf der Symbolleiste **Standard**.
- Es kann helfen die Option ***Drucken im Hintergrund*** unter ***Extras - Optionen - Drucken*** auszuschalten.

Anmerkungen

- Es handelt sich hier um eine sehr oft monierte Schwäche von Word. Kaum befasst sich der Benutzer zum ersten Mal mit etwas komplexeren Word-Dokumenten, sticht ihm diese unsinnige Anzeige in die Augen.
- Um die Gesamtseitenzahl zu referenzieren, also anzuzeigen, greift Word auf eine Feldfunktion zurück. Im ersten Moment gelangt man zur Auffassung, es reiche einfach alle Felder im Dokument zu aktualisieren. Dies liesse sich z.B. mit dem VBA-Befehl: `ActiveDocument.Fields.Update` auf einfache Weise automatisieren. Nun befindet sich in der Praxis aber Seitenzahlen meistens in der Kopf- oder Fusszeile aber genau diese Komponenten werden mit dem genannten Befehl nicht erfasst. Berücksichtigt wird nämlich nur der Hauptteil des Dokumentes.
- Nun kennt Word aber insgesamt nicht weniger als elf Dokumententeile. Diese werden im Objektmodell von Word **StoryRanges** genannt. Allein sechs dieser elf Dokumententeile betreffen Fuss- und Kopfzeilen, wie die folgende Tabelle aufzeigt:

Tabelle der "*StoryTypes*" in Word

Komponente	Word-Objekt	Relation	Bemerkung	Beispiel
Haupttext	MainTextStory	1 pro Dokument	MainTextStory und Dokument können synonym behandelt werden.	`ActiveDocument.Fields.Update`
Hauptkopfzeile	PrimaryHeaderStory	1 pro Abschnitt (Section)	Ist immer vorhanden, kann aber auch leer sein. Kann auch der OddPage-HeaderStory entsprechen, welche es nominell nicht gibt	`ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Fields.Update`
Kopfzeile für erste Seite	FirstPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Erste Seite anders" aktiviert ist.	`ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage).Range.Fields.Update`
Kopfzeile für gerade Seite	EvenPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Gerade-/ungerade Seiten anders" aktiviert ist.	`ActiveDocument.Sections(1).Headers(wdHeaderFooterEvenPages).Range.Fields.Update`
Hauptfusszeile	PrimaryFooterStory	1 pro Abschnitt (Section)	siehe Hauptkopfzeile	`ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range.Fields.Update`
Fusszeile für erste Seite	FirstPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterFirstPage).Range.Fields.Update`
Fusszeile für gerade Seite	EvenPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterEvenPages).Range.Fields.Update`
Kommentare	CommentsStory	0 zu n pro Dokument	Kommentare können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Comments(1).Range.Fields.Update`
Endnote	EndnotesStory	0 oder 1 pro Dokument	Endnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Endnotes(1).Range.Fields.Update`
Fussnoten	FootnotesStory	0 zu n pro Dokument	Fussnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Footnotes(1).Range.Fields.Update`
Textfeld	TextFrameStory	0 zu n pro Story	Textfelder sind in erster Linie Objekte (Elemente) von Shapes	`ActiveDocument.Shapes(1).TextFrame.TextRange.Fields.Update`

Zur programmtechnischen Vereinfachung der Sache, kennt Word die **Next-Story-Range-Methode**. Dank dieser Methode wird ein sequentielles Abarbeiten der Dokumentkomponenten erst möglich. Auf dieser Methode lässt sich ein Makro aufbauen, welches die Aktualisierung der fraglichen Feldern übernehmen kann:

VBA-Code

```
Sub FelderAktualisieren()
  Application.ScreenUpdating = False
  ActiveDocument.Repaginate
  For Each Teil In ActiveDocument.StoryRanges
    Teil.Fields.Update
    While Not (Teil.NextStoryRange Is Nothing)
      Set Teil = Teil.NextStoryRange
      Teil.Fields.Update
    Wend
  Next
  Application.ScreenUpdating = True
End Sub
```

(Die Felder in allen Dokumententeilen werden aktualisiert. Auch die Feldfunktion Seite x von y sollte nach Ablauf des Makros korrekt angezeigt werden, ganz unabhängig davon, ob sich die Feldfunktion im Hauptteil des Dokumentes, oder woanders befindet)

Links zu diese, Thema

Microsoft

Deutsch

► **WD: Feld AnzSeiten wird beim Öffnen einer Datei nicht aktualisiert**

Englisch

► **WD97: NumPages Field Not Updated When You Open File**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SZahlen1.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Word2000: In der Kopf- oder Fusszeile erscheinen die Angaben Seite 1 von 1, Seite 2 von 2, Seite 3 von 3, u.s.w.

◄ Zurück zum Hauptindex

Problem

> **Ich arbeite mit Word2000 und habe in mein Worddokument Seitenzahlen eingefügt und zwar in der Art: Seite 1 von 6, Seite 2 von 6, Seite 3 von 6, u.s.w. In der Seitenlayoutansicht klappt das auch alles. Drucke ich nun aber das Dokument aus, kommt es aber offenbar zu einer Fehlinterpretation. Der Ausdruck weisst nämlich folgende Seitenzahlen aus:**
> ***Seite 1 von 1, Seite 2 von 2, Seite 3 von 3, u.s.w.***

Erklärung

Sie werden vielleicht auch schon festgestellt haben, Word hatte noch nie eine besonders harmonische Beziehung zu Seitenzahlen. Mit Word2000 gesellt sich nun ein zusätzlicher Bug zu denen, welche wir schon aus Word 97 kennen. Dieses Fehlverhalten von Word2000, welches nicht mit allen Druckertreiber auftritt, aber mit vielen, wird mit dem Office2000 Service Release 1 behoben.

Lösung

Installieren Sie die deutschsprachige Version des **Office2000 Service Release 1**. Die Download-Adresse dieses 30 - 50 MB Bug-Fixes ist unten angegeben.

Rundherumläufe

- Aktivieren Sie im Word-Dialog ***Extras - Optionen - Drucken*** die Option **Umgekehrte Druckreihenfolge**. Word sollte sich diese Einstellung bis Widerruf merken
- Als Alternative zum obigen, können Sie sich nach einem neuen Druckertreiber für Ihr Gerät umsehen.

Anmerkungen

- Mit der genannten Einstellung macht der Drucker jetzt vielleicht nicht mehr genau das, was Sie von ihm erwarten. Die letzte Seite wird nun wirklich zuerst gedruckt und das Papier liegt am Schluss in einer anderen Reihenfolge in der Ablage. Immerhin, die Praxis zeigt, die Seitenzahlen werden so korrekt gedruckt.
- Die Art und Weise, wie Sie die Seitenzahlen einfügen, also ob über den Dialog *Seitenzahl einfügen*, über das AutoText-Feature oder über Feldfunktionen, hat keine Auswirkung auf diese Problematik.

Links zu diesem Thema

Microsoft

Deutsch

► **Office2000 Service Release 1a für das deutschsprachige Office**

Englisch

► **WD2000: NUMPAGES Field Prints Incorrect Results**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SZahlen4.htm
René Probst, Oktober 1999

Änderungen

- 17. Juni 2000 - Office2000 Service Release 1a

MS Word startet nur noch ohne Menüleiste, Symbolleisten und Lineal!

◄ Zurück zum Hauptindex

Problemstellung

> **Ich arbeite mit Word2000. Plötzlich nun, wenn ich Word aufstarte fehlt die Menüleiste, sämtliche Symbolleisten und auch das Lineal ist nicht mehr sichtbar. Über *Extras – Anpassen* geht nichts, da auch dieser Befehl nicht mehr erreichbar ist. Eine Neuinstallation von Word hat nichts bewirkt. Was kann ich jetzt tun?**

Erklärung

Dieses Phänomen tritt ordnungsgemäss auf, sollten Sie willentlich oder unabsichtlich in den **Vollbild** Modus geschaltet haben. In diesem Falle können Sie mit der **[Esc]**-Taste wieder in den normalen Modus zurück schalten. Das Problem kann aber auch einen schwerwiegenderer Grund haben.

Beim Beenden von Word, werden die Einstellungen der Symbolleisten, insbesondere die Sichtbarkeit und die Position auf dem Bildschirm, für die nächste Word- Sitzung aufbewahrt. Diese Parameters werden also beim beenden der Anwendung an einem bestimmten Ort in der Registrierdatenbank abgelegt.

Nun ist MS Word leider aber keine stabile Anwendung. Wenn die Anwendung zu einem ungünstigen Zeitpunkt abschmiert, liegen diese Parameters nicht mehr in konsistenter Form in der Registrierdatenbank vor. Word kann beim nächsten Start diese Daten nicht interpretieren und ignoriert diese dann, was zu den beschriebenen Symptomen führt. Die Sache ist um so kritischer, weil sich Microsoft entschlossen hat, diese Parameters gemischt mit vielen anderen Optionen in einer ellenlangen Zeichenfolge abzulegen, was die Sache bestimmt nicht robuster macht.

Gezielt können Sie diese Informationen in der Registrierdatenbank nicht manipulieren, da diese im binärem Format vorliegen. Sie können aber den ganzen Schlüssel löschen, was Word dazu bewegt, beim nächsten Start diese Daten neu zu erstellen. Ein geübter Benutzer erledigt dies binnen wenigen Sekunden. Für Leute, welche sich im Umgang mit der Registrierdatenbank schwer tun, habe ich eine Alternative bereit gestellt. Allgemeine Informationen über den Umgang mit der Registrierdatenbank finden Sie im Link am Schluss dieses Skriptes.

Lösung

Versierte Benutzer

Schliessen Sie Word und löschen Sie folgenden Schlüssel (und zwar den ganzen Ast) in der Registrierdatenbank:

```
HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data
```

Word 97 bzw.

```
HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Data
```

Word2000

Übrige Benutzer

Gehen Sie bitte wie folg vor:

1. **Schliessen** Sie Word.
2. Starten Sie das kleine Programm mit dem Namen **Notepad** (*Start – Programme – Zubehör*).
3. Kopieren Sie die folgenden Zeilen (sie gelten sowohl für Word 97 wie auch für Word2000) in das leere Fenster.
4. Speichern Sie nun die Datei unter dem Namen **WData.reg** ab, wobei eigentlich nur die Dateierweiterung **reg** wesentlich ist.
5. Führen Sie eine Doppelklick auf diese Datei aus.
6. Sie werden nun nach einer Bestätigung gefragt. Bejahen Sie also die Aktion.
7. Sie können Word nun wieder starten. Das Problem sollte behoben sein.

```
REGEDIT4

[-HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data]
[-HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Data]
```

(Diese drei Zeilen speichern wir unter dem Namen WData.reg)

Anmerkungen

- Durch das Löschen des genannten Schlüssels gehen einige Einstellungen, welche Sie allenfalls dann neu vornehmen müssen, verloren. Der Schaden hält sich allerdings in einem recht bescheidenem Umfang, da die meisten Einstellungen in der *Normal.dot* gespeichert werden.

Links zu diesem Thema

Eigene Links

► **Der Umgang mit der Registrierdatenbank und dem Registriereditor**
► **Wo speichert MS Word seine Einstellungen (Umgebungsoptionen)?**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/KeineSL.htm

René Probst, Oktober 2000

Änderungen

- --- nil ---

WD2000: Die Grammatikprüfung lässt sich nicht aktivieren.

◄ Zurück zum Hauptindex

Problem

> **Ich setze seit einiger Zeit Word2000 ein. Will ich über *Extras - Rechtschreibung und Grammatik...* das Dokument auf eine korrekte Grammatik überprüfen geht das nicht. Auch ist mir aufgefallen, dass alle Optionen bezüglich der Grammatikprüfung im Dialog *Extras - Optionen... - Rechtschreibung und Grammatik* ausgegraut und damit nicht erreichbar sind.**

Voranbemerkungen

Das Problem ist bekannt. Es scheint recht oft aufzutreten nachdem Office2000 (neu) installiert wurde. Ich habe Hinweise darauf, dass es ausschliesslich dann passieren soll, wenn Office2000 über Office 97 installiert wird.

Microsoft sagt aus , dass das Problem nach der Installation des Office2000-Service Releases behoben ist.

Wie auch immer, das folgende Vorgehen führt mit oder ohne Service Release zum Ziel. Ich habe die Methode in enger Zusammenarbeit mit der Microsoft *Help Desk* in München entwickelt. Mittlerweile hat es sich dutzendfach bewährt. Wahrscheinlich liesse sich die Sache abkürzen. In Angesicht des grossen Erfolges (keine negativen Rückmeldungen) habe ich aber bewusst nie etwas daran geändert.

Lösung

Zusammengefasst sollten Sie folgendes tun:

- Korrekturhilfen deinstallieren
- Das *Proof*-Verzeichnis umbenennen
- Zwei Schlüssel in der Registrierdatenbank löschen
- Korrekturhilfen wieder installieren

Gehen Sie im einzelnen wie folgt vor:

Achtung: Das Vorgehen bezieht sich auf Word2000, bzw. Office2000

1. Rufen Sie den Dialog Start - Einstellungen - Systemsteuerung - Software - Microsoft Office2000 auf.
2. Betätigen Sie den Schaltknopf mit der Bezeichnung **Hinzufügen/Entfernen**
3. Betätigen Sie nun den Schaltknopf mit der Bezeichnung **Features hinzufügen/entfernen**.
4. Unter *Office Tools* finden Sie den Eintrag **Korrekturhilfen**. Klicken Sie auf diesen Eintrag und wählen Sie im Kontextmenü **nicht verfügbar**.
5. Betätigen Sie nun den Schaltknopf **Jetzt aktualisieren** und schliessen Sie den Office Setup ab.
6. Starten Sie nun den Rechner neu.
7. Lassen Sie nach einem Unterverzeichnis mit dem Namen **Proof** suchen und nennen Sie dieses in *ProofAlt* um. Dieses Verzeichnis wird später dann neu erstellt.
8. Starten Sie nun den Registriereditor und löschen Sie folgende beiden Schlüssel: (jeweils den ganzen Ast löschen)

   ```
   HKEY_CURRENT_USER\Software\Novell\Grammatik
   HKEY_LOCAL_MACHINE\Software\Novell\Grammatik
   ```

   Diese Schlüssel werden später neu erstellt. Der Name *Novell* ist kein Irrtum. Die Grammatikprüfung stammt ursprünglich von dieser Firma.
9. Installieren Sie nun die Korrekturhilfen über den Office-Setup neu.

Das Problem ist damit gelöst.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Gramm.htm

René Probst, Oktober 2000

Änderungen

- --- nil ---

Wie kann ich in allen Dokumententeile, also auch in Kopf- und Fusszeile sowie in Textfeldern die Felder aktualisieren?

◄ Zurück zum Hauptindex

Problem

> **Mit dem Word-Befehl *Felder aktualisieren* werden mir zwar die Felder im Hauptteile des Dokumentes aktualisiert, Kopf- und Fusszeilen sowie Textfeldern sind davon aber leider nicht betroffen.**

Antwort

Diese Beobachtung ist richtig. Das gleiche werden Sie übrigens auch feststellen, wenn Sie im VBA-Code den Befehl `ActiveDocument.Fields.Update` verwenden.

Lösung

- Wechseln Sie in die Kopf- Fusszeilenansicht. Nun können Sie diesen Bereich markieren und den Befehl ***Felder aktualisieren*** nochmals aufrufen.
- Alternativ können Sie auch in die **Seitenansicht** wechseln und wieder zurück in die Normalansicht. Die Schaltfläche um in die Seitenansicht zu wechseln, finden Sie üblicherweise auf der Symbolleiste **Standard**.

Anmerkung

MS Word kennt insgesamt nicht weniger als elf Dokumententeile; in der Word-Sprache **Story Ranges** genannt. Gut die Hälfte davon, nämlich sechs, betreffen Kopf- und Fusszeilen. Die nachfolgende Tabelle, weist diese elf Dokumententeile aus. Für Leute, welche VBA programmieren, ist es gut zu wissen, dass es eine Methode gibt, welche es erlaubt, diese Dokumententeile sequentiell abzuarbeiten. Damit wird es möglich, einen allgemeingültigen Code zu schreiben, welcher sich auf alle Dokumententeile bezieht und dies, ohne im Voraus wissen zu müssen, welche Teile im Dokument auch tatsächlich vorkommen.

Tabelle der «*StoryTypes*» in Word

Komponente	Word-Objekt	Relation	Bemerkung	Beispiel
Haupttext	MainTextStory	1 pro Dokument	MainTextStory und Dokument können synonym behandelt werden.	`ActiveDocument.Fields.Update`
Hauptkopfzeile	PrimaryHeaderStory	1 pro Abschnitt (Section)	Ist immer vorhanden, kann aber auch leer sein. Kann auch der OddPage-HeaderStory entsprechen, welche es nominell nicht gibt	`ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Fields.Update`
Kopfzeile für erste Seite	FirstPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Erste Seite anders" aktiviert ist.	`ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage).Range.Fields.Update`
Kopfzeile für gerade Seite	EvenPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Gerade-/ungerade Seiten anders" aktiviert ist.	`ActiveDocument.Sections(1).Headers(wdHeaderFooterEvenPages).Range.Fields.Update`
Hauptfusszeile	PrimaryFooterStory	1 pro Abschnitt (Section)	siehe Hauptkopfzeile	`ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range.Fields.Update`
Fusszeile für erste Seite	FirstPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterFirstPage).Range.Fields.Update`
Fusszeile für gerade Seite	EvenPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterEvenPages).Range.Fields.Update`
Kommentare	CommentsStory	0 zu n pro Dokument	Kommentare können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Comments(1).Range.Fields.Update`
Endnote	EndnotesStory	0 oder 1 pro Dokument	Endnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Endnotes(1).Range.Fields.Update`
Fussnoten	FootnotesStory	0 zu n pro Dokument	Fussnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Footnotes(1).Range.Fields.Update`
Textfeld	TextFrameStory	0 zu n pro Story	Textfelder sind in erster Linie Objekte (Elemente) von Shapes	`ActiveDocument.Shapes(1).TextFrame.TextRange.Fields.Update`

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub FelderAktualisieren()
  Application.ScreenUpdating = False
  ActiveDocument.Repaginate
  For Each Teil In ActiveDocument.StoryRanges
    Teil.Fields.Update
    While Not (Teil.NextStoryRange Is Nothing)
      Set Teil = Teil.NextStoryRange
      Teil.Fields.Update
    Wend
  Next
  Application.ScreenUpdating = True
End Sub
```

(Die Felder in allen Dokumententeilen [Story Ranges] werden aktualisiert.)

☹ Wegen eines Fehlverhaltens von MS Word, wird der Text von Textfeldern, wenn sich letztere in der Kopf- oder Fusszeile befinden, nicht erfasst. Siehe dazu den Link zur Datenbank von Microsoft am Ende dieses Skriptes.

Links zu diesem Thema

Microsoft

► WD2000: StoryRanges Does Not Return Text Box Text in Header/Footer

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Feldera.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich mit MS Word nicht zusammenhängende Textstellen im Dokument gleichzeitig markieren?

◄ Zurück zum Hauptindex

Frage

> Ich bin gerade dabei, meinem Word-Dokument den letzten Feinschliff zu verleihen. Dabei taucht die Frage auf, welche Möglichkeit ich habe, mehrere nicht zusammenhängende Textstellen im Dokument gleichzeitig zu markieren, um diese anschliessend mit einem Streich zu formatieren.

Antwort

Es ist in MS Word leider nicht möglich, nicht zusammenhängende Textstellen gleichzeitig zu markieren, auch nicht unter Einsatz von VBA. Die Architektur von MS Word lässt dies einfach nicht zu.

Der Vergleich, welcher manchmal mit Excel gezogen wird, wo es ja unter Einsatz einer Tastenkombination sehr einfach möglich ist, nicht zusammenhängende Zellen oder Zellbereiche gleichzeitig zu markieren, hinkt, denn bei Excel-Zellen handelt es sich um Objekte, nicht aber bei Textpassagen in einem Word-Dokument. **Text fliesst nun mal.** Handelt es sich in Word um Objekte zum Beispiel um eingefügte Grafiken, ist eine Mehrfachmarkierung auch möglich. Halten Sie dazu einfach die **[Umschalt]** Taste gedrückt.

Alternativen

Zum Glück erweist sich MS Word letztlich aber doch als vielseitig und es lassen interessante Alternativen zur Mehrfachmarkierung finden.

Der Formatpinsel

Auf der Symbolleiste mit der Bezeichnung **Standard** finden Sie den **Formatpinsel**, auch **Format Painter** genannt.

Viele Anwender wissen, dass es damit möglich ist, die Formatierung von einer Stelle auf eine andere Stelle im Dokument zu übertragen. Gehen Sie dabei wie folgt vor:

1. Stellen Sie die Einfügemarke an eine Stelle im Text, welche bereits die gewünschte Formatierung aufweist.
2. Klicken Sie nun den **Formatpinsel** an. Dabei nimmt der Zeiger die Form eines malenden Pinsels an.
3. Markieren Sie nun die Textstelle, welche mit dem *aufgezogenen* Format, formatiert werden soll.
4. Sind Sie so vorgegangen verliert die Einfügemarke das besondere Aussehen und ihre magische Funktion wieder.

Was schon weniger bekannt ist, ist die Tatsache, dass der Doppelklick auf den Formatpinsel auch eine Funktion hat und zusätzliche Möglichkeiten eröffnet. Der Clou beim Doppelklick ist, dass das magische Verhalten des Mauszeigers auch nach einer Markierung erhalten bleibt. So ist es möglich, das *aufgezogene* Format hintereinander auf beliebig viele nicht zusammenhängende Textstellen zu übertragen. Dies stellt eine effiziente Art des Formatierens dar.

Drücken Sie die **[Esc]**-Taste, um diese Zauberei zu beenden. Der Mauszeiger nimmt dann wieder die übliche Gestalt an.

Hervorheben und dann umwandeln

Eine weitere Schaltfläche, welche in diesem Zusammenhang eingesetzt werden kann, ist der Schaltknopf mit dem Namen **Hervorheben** auf der Symbolleiste mit der Bezeichnung **Format**. Mir persönlich geht die folgende Methode fast besser von der Hand, als jene mit dem Formatpinsel. Gehen Sie konkret wie folgt vor:

1. Führen Sie einen **Doppelklick** auf das Symbol **Hervorheben** aus. Der Mauszeiger verwandelt sich dadurch in einen Leuchtstift.
2. Markieren Sie nacheinander jene Textstellen, welche eine bestimmte neue Formatierung erhalten sollen. Diese Stellen werden damit farbig hinterlegt.
3. Rufen Sie nun vom Word-Hauptmenü den Dialog ***Bearbeiten - Ersetzen...*** auf.
4. Während die Einfügemarke im Feld **Suchen nach:** steht, klicken Sie auf diesem Dialog folgendes an: ***Erweitern - Format - Hervorheben***.
5. Platzieren Sie nun die Einfügemarke in das Feld **Ersetzen durch:**
6. Klicken Sie nun auf ***Format - Zeichen...*** oder auf ***Format - Formatvorlage...*** und wählen Sie aus, wie die hervorgehobenen Stellen neu formatiert werden sollen.
7. Betätigen Sie nun den Schaltknopf **Alle Ersetzen**.
8. Markieren Sie nun das gesamte Dokument mit der Tastenkombination **[Alt + A]**.
9. Wählen Sie nun im *Popup*-Menü beim Schaltknopf **Hervorheben** die Farbe **Keine**.

Formatierten Text mit VBA eingeben

Mit unter taucht der Wunsch auf, formatierter Text mittels VBA in ein Word-Dokument einzufügen. Soweit dieser formatierter Text aus einem anderen Word-Dokument stammt, sind Sie gut bedient, verwenden Sie dazu die Eigenschaft *FormattedText*.

Sollte dieser formatierter Text nun aber von einem anderen Ort stammen, dann wird es wirklich umständlich, ist MS Word doch eine format-orientierte Anwendung, wobei dieses Format nun mal völlig proprietär (d. h. keinem offiziellen Standard folgend) ist. Mit Text, welcher mit sogenannten *Tags* formatiert ist, wie wir es z.B. von HTML kennen, wäre die Sache natürlich wesentlich einfacher.

Ich schlage vor, trotzdem den Text mit solchen Tags in einem ersten Schritt in das Word-Dokument zu übergeben und dann in einem zweiten Schritt, den so übergebenen Text mit der sehr starken Suchen/Ersetzen-Funktion von MS Word in formatierten Word-Text umzuwandeln.

Es ist zwar nicht zwingend, allerdings spricht wenig dagegen, als Vorgabe in etwa das zu applizieren, was uns HTML vormacht. Wie gesagt, zwingend ist dass nicht, Sie können sich selbstverständlich auch eine andere Konvention ausdenken. In diesem Sinne schlage ich vor, wir geben - z.B. in einer VBA-Variablen - folgenden Text, welcher mit Tags garniert ist vor:

```
Das nächste Wort ist $b$fett$/b$ geschrieben und das folgende Wort ist $i$kursiv$/i$
geschrieben. Das nächste Wort ist $b$fett$/b$ geschrieben und das folgende Wort ist
$i$kursiv$/i$ geschrieben. Das aller letzte Wort ist $b$$i$fett und kursiv$/b$$/i$
geschrieben.
```

Die VBA-Routine, um daraus Word-formatierten Text zu machen könnte wie folgt aussehen, wobei das Ergebnis sich wie folgt präsentiert:

Das nächste Wort ist **fett** geschrieben und das folgende Wort ist *kursiv* geschrieben. Das nächste Wort ist **fett** geschrieben und das folgende Wort ist *kursiv* geschrieben. Das aller letzte Wort ist ***fett und kursiv*** geschrieben.

```
Sub StringNachFormatiertenText()
  tStr = "Das nächste Wort ist $b$fett$/b$ geschrieben und das folgende "
  tStr = tStr & "Wort ist $i$kursiv$/i$ geschrieben."
  tStr = tStr & " " & tStr
  tStr = tStr & " Das aller letzte Wort ist $b$$i$fett + kursiv$/b$$/i$ geschrieben."
  Dim oRange As Range
  Set oRange = Selection.Range
  oRange.InsertAfter tStr
  oRange.SetRange Start:=oRange.Start, End:=oRange.Start + Len(tStr)
  TagsNachFormat oRange, "b", True, False
  TagsNachFormat oRange, "i", False, True
End Sub

Private Sub TagsNachFormat(oRange As Range, Tag As String, _
   Fett As Boolean, Kursiv As Boolean)
  With oRange.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Format = True
    .MatchWildcards = True
    .Text = "$" & Tag & "$*$/" & Tag & "$"
    If Fett = True Then .Replacement.Font.Bold = True
    If Kursiv = True Then .Replacement.Font.Italic = True
    .Execute Replace:=wdReplaceAll
    .MatchWildcards = False
    .Format = False
    .Text = "$" & Tag & "$"
    .Replacement.Text = ""
    .Execute Replace:=wdReplaceAll
    .Text = "$/" & Tag & "$"
    .Replacement.Text = ""
    .Execute Replace:=wdReplaceAll
  End With
End Sub
```

(Text aus einer String-Variablen, welcher mit *Tags* versehen ist, wird in das Dokument eingebracht und dann in Word-formatierten Text umgewandelt.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Select.htm

René, Probst, Februar 2001

Änderungen

- --- nil ---

Rechtschreibprüfung und andere Funktionen stehen in einem teilweise geschützten Dokument nicht zur Verfügung

◄ Zurück zum Hauptindex

Problem

> Ich habe ein Formular, welches ich in Abschnitten unterteilt habe. Ein Abschnitt davon habe ich bewusst geschützt, alle anderen Abschnitte sind nicht geschützt. Trotzdem funktioniert auch in diesen nicht geschützten Abschnitten die Rechtschreibkennzeichnung und andere Funktionen nicht. Erst nach der Aufhebung des Schutzes für das gesamte Dokument, ist Word wieder voll funktionsfähig. Die Aufhebung des Formularschutzes bzw. dessen nachträgliche Reaktivierung, hat aber zur Folge, dass sämtliche Eingaben in den Formularfeldern verloren gehen.

Antwort

Tatsächlich reicht es, einen einzigen Abschnitt gegen Überarbeitung, Kommentare oder Formulareingabe zu schützen und schon stehen sehr viele Funktionen von Word auch in den ungeschützten Abschnitten nicht mehr zur Verfügung. Es handelt sich dabei um eine sehr deutliche Schwäche von MS Word, welche die Option, nämlich nur einzelne Abschnitte in einem Word-Dokument zu schützen, ziemlich zur Farce macht.

Interessanterweise waren die Word-Versionen 6.0 und 7.0 (95) hier weniger restriktiver. In **Word2000** (und nur in Word2000) funktioniert die interaktive Rechtschreibprüfung **nachdem der Office2000 Service Release 1** installiert wird. Alle anderen Funktionen, welche unten aufgeführt werden, sind allerdings weiterhin nicht zugänglich.

Funktionen, welche in einem teilweise geschütztem Dokument auch in ungeschützten Abschnitten nicht zur Verfügung stehen

Vorbemerkend sei gesagt, es wäre fast einfacher aufzuzählen, was denn überhaupt noch zur Verfügung steht. Die Einschränkung des Funktionsumfanges hängt auch etwas von der Art des Schutzes ab, also ob das Dokument gegen Überarbeitung, gegen Kommentare, oder gegen Formulareingaben geschützt wird.

- Interaktive Rechtschreibung (exkl. Word2000 mit Office2000 Service Release 1)
- Silbentrennung
- Gliederungsansicht
- Kopf- und Fusszeilenansicht
- Einfügen von Textmarken
- Einfügen von Verweisen
- Einfügen von Beschriftungen
- Einfügen von Positionsrahmen
- Nummerierung und Aufzählungen
- Rahmen und Schattierungen
- Anpassen von Symbolleisten
- Seriendruck
- Verändern von Dokumenteneigenschaften
- etc.

Lösung

Falls Sie mit Word2000 arbeiten und es Ihnen um die interaktive Rechtschreibprüfung geht, installieren Sie den Office2000 Service Release 1. Die Download Adresse für diesen sehr grossen Patch, ist unten angegeben.

Mittels VBA-Code in Word 97 und Word2000, (ebenfalls in Word 6.0 dort aber noch mit Word Basic) ist es immerhin möglich, den Dokumentenschutz gezielt aufzuheben um anschliessend die nötigen Aktionen durchzuführen. Zum Schluss hin, sollte das Makro den Schutz wieder aktivieren. Ebenfalls mittels VBA ist es möglich, den Dokumentenschutz in einer Art und Weise zu setzen, dass die Formulareingaben des Benutzers **nicht** verloren gehen.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

Dynamisches Aufheben des Formularschutzes

Der nachfolgende Code hebt den Formularschutz eines Dokumentes, welches über mindestens zwei Abschnitte verfügen sollte, dynamisch auf, um die aktuelle Zeit in die Fusszeile des zweiten Abschnittes einzufügen. Zum Schluss wird der Schutz wieder aktiviert. Die Formularfeldeingaben bleiben dabei erhalten.

```
Sub FormularschutzDynamischAufheben()
  Dim oDoc As Document
  Dim oRange As Range
  Set oDoc = ActiveDocument
  If oDoc.ProtectionType >= 0 Then oDoc.Unprotect
  Set oRange = oDoc.Sections(2).Footers(1).Range
  oRange.Fields.Add Range:=oRange, Type:=wdFieldTime
  oRange.InsertBefore "Ausgefüllt um: "
  oDoc.Protect Type:=wdAllowOnlyFormFields, Noreset:=True
End Sub
```

(Beispiel welches die dynamische Aufhebung des Formularschutzes demonstriert)

Ein-/Ausschalter für den Formularschutz ohne Verlust der Benutzereingabe

Nachfolgend wird beschrieben, wie Sie in der Formularvorlage (Dokumentenvorlage) einen **Ein/Aus-Schalter** einrichten können, welchen den Formularschutz setzt bzw. aufhebt, die Formulareingaben des Benutzers dabei aber intakt lässt.

Bei der Einrichtung dieses Features gehen Sie bitte wie folgt vor:

1. **Öffnen** Sie die Dokumentenvorlage, welches die Formularfelder festlegt.
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
3. Kopieren Sie den gesamten Code, also beide Prozesse in ein neues Modul.
4. Lassen Sie das Makro mit dem Namen **FormularSchutzToggleKnopfEinrichten** ausführen. Dieser Prozess muss grundsätzlich nur einmal ablaufen.
5. Sie sollten nun auf der Symbolleiste **Standard** eine neue Schaltfläche vorfinden. Sie erkennen diese Schaltfläche am *Smily-Symbol*.

Anmerkungen

- Sie können diese Schaltfläche nach belieben über ***Extras - Anpassen*** verschieben und Ihren eigenen Wünschen anpassen.
- Die zweite Prozedur mit dem Namen **FormularSchutzToggleKnopfProzess** wird jedes Mal aufgerufen, wenn Sie den neuen Knopf betätigen. Falls Sie mehrere Formularvorlagen besitzen, können Sie diesen Prozess auch in der **Normal.dot** aufnehmen statt in jeder Formularvorlage.
- Der neue Knopf hat ein kleines Manko. Wenn Sie den Formularschutz nun doch über den entsprechenden Word-Dialog aufrufen, synchronisiert sich der Knopf nicht. Konkret heisst dies, er verändert sein Aussehen nicht und wird als ausgeschaltet angezeigt, obwohl der Schutz eingeschaltet ist oder v.v. Beim nächsten Gebrauch des Knopfes synchronisiert er sich allerdings wieder.

```
Sub FormularSchutzToggleKnopfEinrichten()
  Application.CustomizationContext = NormalTemplate
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars("Standard").Controls.Add(ID:=1)
  With Knopf
    .Caption = "Formularschutz aktivieren"
    .TooltipText = "Schaltet den Formularschutz ein"
    .OnAction = "FormularSchutzToggleKnopfProzess"
    .FaceId = 277
  End With
End Sub

Private Sub FormularSchutzToggleKnopfProzess()
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars.ActionControl
  With Knopf
    Select Case ActiveDocument.ProtectionType
      Case wdNoProtection
        ActiveDocument.Protect Type:=wdAllowOnlyFormFields, NoReset:=True
        .State = msoButtonUp
        .Caption = "Formularschutz aufheben"
        .TooltipText = "Hebt den Formularschutz auf"
      Case wdAllowOnlyRevisions, wdAllowOnlyComments
        MsgBox "Das Dokument stellt kein gültiges Formular dar."
        Exit Sub
      Case wdAllowOnlyFormFields
      ActiveDocument.Unprotect
      .State = msoButtonDown
      .Caption = "Formularschutz aktivieren"
      .TooltipText = "Schaltet den Formularschutz ein"
    End Select
  End With
End Sub
```

(Ein-Ausschalter für den Formularschutz. Im Gegensatz zum Word-integrierten Schalter, bleiben die Formularfeldinhalte hier erhalten.)

Verändern von Dokumenteneigenschaften bei geschützten Formularen

Wie schon angedeutet, können Sie die Dokumenteneigenschaften (***Datei - Eigenschaften*** in einem Formular nicht ändern, wenn das Formular geschützt ist. Hier gibt es allerdings eine Hintertür.

Über den **Windows-Explorer** geht diese Sache problemlos. Führen Sie einen Rechtsklick auf die fragliche Datei aus, wählen Sie im Kontextmenü **Eigenschaften** und wechseln Sie in das Register **Datei-Info**. Nun können Sie die gewünschten Änderungen vollbringen.

Links zu diesem Thema

Microsoft

Deutsch

► **Wd97: Einige Menübefehle im geschützten Dokument nicht verfügbar**

► **Downloaddetails zu Office2000 Service Release 1a (SR-1a) Update**

Englisch

► **WD97: Some Menu Commands Unavailable (Document Protection)**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/DSchutz.htm

René Probst, Oktober 1999

Änderungen

- 10. November 1999 - Ein-/Ausschalter
- 12. Dezember 1999 - Dokumenteneigenschaften über Windows-Explorer ändern
- 19. Juni 2000 - Office2000 SR-1a

Wo kann ich das Wörterbuch zur neuen deutschen Rechtschreibung beziehen?

◄ Zurück zum Hauptindex

Problem

> Die neue Rechtschreibung ist bzw. wird in der nächsten Zeit sowohl in Deutschland wie auch in Österreich und in der Schweiz verbindlich. Produkte, wie z.B. Word 6.0, Word 95 und Word 97, beinhalten aber nur die Rechtschreibetools für die alte deutsche Rechtschreibung.

Lösung

Word 6.0

Es gibt seit längerer Zeit einen Ersatz für das ursprünglich ausgelieferte Wörterbuch. Dieses neue Wörterbuch taxiert sowohl die alte, wie auch die neue deutsche Rechtschreibung als korrekt. Der Nachteil dieser Lösung liegt auf der Hand: Sie sollten sich in Zukunft nicht wundern, wenn Ihnen Texte unter die Augen kommen, welche ein buntes Müsli zwischen alt und neu darstellen.

Bei der Word Version 6.0 handelt es sich noch nicht um eine reine 32-Bit-Anwendung. Microsoft plant keine weiteren Anstrengungen mehr bezüglich der Rechtschreibtools für Word 6.0.

Sie erfahren weiter unten, wo Sie dieses Wörterbuch mit dem Dateinamen **MSSP2_GE.LEX** beziehen können. Um es zu aktivieren, reicht es, das alte Wörterbuch, welches den selben Dateinamen aufweist, mit der neuen Datei zu überschreiben. Der Download umfasst knapp 0,5 MB. Der Link ist weiter unten eingeblendet.

Word 95 und Word 97

Sie finden das Wörterbuch für die neue deutsche Rechtschreibung auf dem deutschen Server von Microsoft. Es ist für die Versionen Word 95 (also Word 7.0) und Word 97 (also Word 8.0) einsetzbar.

Nach erfolgreichem Download und Installation, finden Sie im **Startmenü** unter **Programme - Neue Rechtschreibung**, die Möglichkeit zwischen alt und neu umzuschalten.

Dieses Tool ersetzt eine frühere Version von Microsoft, welches sowohl die alte wie auch die neue Rechtschreibung als richtig taxierte und so ein Müsli zuliess.

Das Wörterbuch samt Installationsroutine hat stattlich an Grösse zugenommen. Der Download umfasst jetzt mehr als 7MB. Der Link ist weiter unten eingeblendet.

Word2000

Falls Sie Office2000 einsetzen ist die neue Rechtschreibung (sowie auch die alte) bereits im Lieferungsumgang enthalten. Sie können vom Word-Hauptmenü den Dialog ***Extras - Optionen - Rechtschreibung*** aufrufen und die Option ***Neue deutsche Rechtschreibung*** ein oder ausschalten. Word2000 sollte sogar erkennen, nach welchen Regeln Sie schreiben bzw. welchen Regeln das Dokument folgt. Dieser Automatismus hat aber so seine Tücken. Wenn Sie oft technische Abhandlungen mit englischen Begriffen schreiben, kommt es vor, dass Word gegen Ihren Willen auf Englisch umstellt. Ich habe deshalb die automatischer Erkennung ausgeschaltet, und fahre so besser.

! Das Wörterbuch von Office2000 ist **nicht** rückwärtskompatibel.

Einschränkungen und Mängel

Allgemein

Die Wörterbücher entfalten ihre ganze Funktionalität nur in MS Word. Für andere Office-Anwendungen gelten diverse Einschränkungen. Die alternative Rechtschreibung steht z. B. in der nicht ganz harmonischen Office2000 Familie nur in Word zur Verfügung.

Word 6.0

Es gibt keine Lösung, welche es erlauben würde, das Umfeld so einzustellen, dass entweder nur die alte Schreibweise oder aber die neue Schreibweise zum tragen käme.

Word 95 und Word 97 sowie Word2000

Die neuen Rechtschreibtools hinterlassen einen zwiespältigen Eindruck. Einerseits habe ich zur Prüfung mal das Verhalten von Word im Eilzugstempo mit den Regeln, welche dpa (Deutsche Presse-Agentur GmbH) herausgegeben hat, verglichen. Dabei hat z. B. Word2000 erstaunlich gut abgeschnitten. Bei der täglichen Arbeit stösst man aber immer mal wieder auf erhebliche Mängel und Schwächen. Ich möchte hier nicht unerwähnt lassen, dass es mir bewusst ist, dass ein gutes und praxistaugliches Rechtschreibetool im Umfeld der Software-Entwicklung auf die Stufe von absolute *High-tech* zu setzen ist. Die Sache lässt sich eben eher schlecht parametrisieren und systematisch anpacken.

- Das Vorzeigebeispiel für die neue deutsche Rechtschreibung, nämlich **Delphin** wird trotz eingestellter neuer Rechtschreibung als korrekt taxiert.
- Auch bei einfach zusammengesetzten Wörtern kommt es sehr oft vor, dass wirres Zeugs, welches eindeutig auf einem Vertippen beruht, leider als korrekt eingestuft wird. Beispiele dafür sind:
  - Kündigungsschatz
  - Grundhage
  - Versadler

Jetzt soll mir bitte jemand erklären, was ein Versadler ist?!

- Viele Wörter, welche nach den neuen Rechtschreibregeln getrennt geschrieben werden müssen, werden von Word auch in der zusammengeschriebenen Version immer noch als korrekt taxiert. Hier eine kleine Liste, welche sich fast beliebig erweitern liesse:
  - aneinanderfügen
  - auseinanderbiegen
  - vorwärtskommen
  - krebserregend
  - übriglassen
  - wiedersehen

Hier irritiert Word den Umsteiger also mehr, als konkrete Hilfe anzubieten.

- Mit unter erreicht Word nicht mal Grundschulniveau. Hier einige Beispiele:
  - **Wiederstand** wird **nicht** moniert.
  - **Wiederspruch** wird **nicht** moniert.
  - **Endgeld** wird **nicht** moniert.

Keine grosse Hilfe also, für Leute, welche sich mit der deutschen Rechtschreibung schwer tun.

- Wörter, welche meiner Meinung nach korrekt sind, werden von Word2000 moniert. Keck schlägt Word dann aber Alternativen vor, welche einen sprachlichen Unsinn darstellen. Beispiele dafür sind: (Auch diese Liste liesse sich sicher noch ergänzen)
  - Ursprungsort: wird als falsch taxiert. Word schlägt stattdessen Ursprungsrot vor.
  - referenzieren: wird als falsch taxiert. Word schlägt stattdessen referenzirren vor. (Wer irrt wohl hier?)

- Bei komplexen Dokumenten, welche oft zwischen Textteile mit Prüfung bzw. ohne Prüfung wechseln, steigt die interaktive Rechtschreibprüfung (Rechtschreibung während der Eingabe prüfen) ganz schlicht und einfach aus, d.h. es wird gar nichts mehr rot markiert. Ruft man die Rechtschreibprüfung über den Dialog *Extras - Rechtschreibung und Grammatik...* auf, wird man mit folgendem Spruch beglückt:
  Es sind zu viele Rechtschreibfehler vorhanden um angezeigt zu werden.....
  Dieser Spruch erscheint leider auch dann, wenn alle Passagen, welche nicht viel mit der deutschen Sprache zu tun haben (also technische Sachen) explizit auf "nicht prüfen" gesetzt wurden. Dieses Verhalten habe ich in Word 97 nie beobachtet. Es stört sehr bei der Arbeit, muss ich doch jetzt einzelne Textpassagen in ein neues Dokument kopieren, damit eine Anzeige der Rechtschreibfehler erscheint. *(Where do you want to go today?* - Ihre Microsoft).
- Hat man die Option ***Extras - Optionen... - Rechtschreibung und Grammatik - Grammatik während der Eingabe prüfen*** aktiviert, kommt es immer mal vor, dass Word2000 z.B. beim Blättern **abschmiert**. Das Betriebssystem aus dem selben Software-Haus geht dann zu allem Unglück auch noch in die Knie. Neben dem Datenverlust ist also noch ein Neustart des Rechners fällig. Sie fahren in der Praxis besser, wenn Sie diese Option generell abschalten, zu mindest bis der Office Service Release 1 für Office2000 erscheint, welcher nach Aussage Microsoft hier Linderung verspricht.
- Es gilt leider weiterhin, dass die Rechtschreibprüfung in Dokumenten, welche (teilweise) geschützt sind, nicht zur Verfügung steht und zwar auch nicht in jenen Abschnitten, welche bewusst nicht geschützt wurden.

Konvertierung bestehender Dokumente von der alten Rechtschreibung nach der neuen

Wenn Sie planen Ihre bestehenden Dokumente von der alten Rechtschreibung nach der neuen Rechtschreibung zu konvertieren, dann können Sie - gratis - eine Software, welche das deutsche PC-Magazin **c't** entwickelt hat über das Internet beziehen. Dieses Programm, welches in VBA programmiert ist, soll allerdings viel Ressourcen binden und langsam sein. Das gleichen Download-Paket enthält auch eine Lösung für die interaktive Prüfung von Word-Dokumenten nach den neuen deutschen Rechtschreiberegeln. Nach der Veröffentlichung der Lösung von Microsoft, ist dieser Teile allerdings obsolet.

Auch der Duden-Verlag soll ein Software-Paket anbieten, mit dem es unter anderem möglich ist, Dokumente, welche mit der alten Schreibweise erstellt wurden, in die neue Schreibweise umzuwandeln.

Links zu diesem Thema

Microsoft

Word 95 und Word 97

► **Neue deutsche Rechtschreibung für Microsoft Office 95 und Microsoft Office 97 (12. April 2000)**

Word 6.0

► **Rechtschreibreform für Word (Word 6.0)**

PC-Magazin c't

► **Tausche Alt gegen Neu**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Rechtsch.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wo kann ich Wörterbücher für Fremdsprachen beziehen?

◄ Zurück zum Hauptindex

Problem

> Mitunter schreibe ich Briefe in spanischer Sprache. Dabei wäre es sehr hilfreich, wenn ich auf die Rechtschreib- und Grammatikprüfung für Spanisch zurück greifen könnte. Was muss ich dazu tun, damit MS Word das kann?

Vorbemerkungen

Module für verschiedene Sprachen sind auf der Office 97-CD bzw. Office2000-CD bereits enthalten. Schreiben Sie Texte in diesen Sprachen reicht es also, die Rechtschreibtools von der CD-ROM zu installieren. Nachdem Sie dies getan haben können Sie über den Word-Dialog ***Extras - Sprache - Sprache bestimmen...*** einem markierten Text eine Sprache zuweisen. Selbstverständlich können Sie auch Formatvorlagen erstellen und dort die Sprache vorgeben.

Welche Komponenten beinhalten die Rechtschreibetools für Office 97 und Office2000?

Ein kompletter Satz für die Rechtschreibetools umfasst folgende Komponenten:

- Ein **Wörterbuch** Für die Rechtschreibung. Der Name der Datei ist wie folgt aufgebaut: **MSSPv_Sprachcode.LEX** also z.B. *MSSP2_GE.LEX*.
- Eine Datei für die **Grammatik** Der Name der Datei ist wie folgt aufgebaut: **MSGR_Sprachcode.LEX** also z.B. *MSGR_GE.LEX*.
- Eine Datei für die **Silbentrennung** Der Name der Datei ist wie folgt aufgebaut: **MSHY_Sprachcode.LEX** also z.B. *MSHY_GE.LEX*.
- Eine Datei für den **Thesaurus** (enthält Synonyme) Der Name der Datei ist wie folgt aufgebaut: **MSTH_Sprachcode.LEX** also z.B. *MSTH_GE.LEX*.
- Je nach Sprache eine unterschiedliche Anzahl von Programmbibliotheken (DLL's).

Die hier aufgeführten Dateinamen sind nur als eine Richtlinie zu verstehen. Je nach Sprachmodul kann die vom Hersteller verwendete Konvention leicht variieren. Sie sollten hier etwas intuitiv vorgehen.

Die meisten dieser Komponenten scheinen auch unabhängig voneinander zu funktionieren. Allerdings ist es wichtig, dass auch die nötigen Programmbibliotheken (**DLL's**) vorhanden sind. Diese Programmbibliotheken registrieren sich in gewissen Fällen automatisch, in anderen Fällen müssen sie technisch registriert werden bzw. in die Registrierdatenbank eingetragen werden. Dies wird in der Praxis oft vergessen bzw. unterschätzt.

Wie installiere ich die Rechtschreibetools für Fremdsprachen?

- Die Sprachtools welche im Lieferungsumfang Ihres Office 97 oder Office2000 enthalten sind, installieren Sie über den **Office-Setup**.
- Die Sprachtools, welche Sie von der Firma **Alki Software Corporation** erwerben installieren sich automatisch per Mausklick. Microsoft selbst vertreibt keine ergänzenden Sprachtools für Office 97 Anders was Office2000 betrifft. Für diese Version von Office gibt es von Microsoft eine CD-ROM mit dem Titel **Office2000 Proofing Tools**. Diese Produkt muss in aller Regel bei Bedarf extra erworben werden.
- Falls der Automatismus der Installationsroutinen in Office 97 aus irgend einem Grund versagen sollte, ist es wichtig zu wissen, dass es in der Regel reicht, die nötigen Dateien in das richtige Verzeichnis zu kopieren. Auf einer typischen Installation, welche ursprünglich auf Deutsch erfolgte, heisst dieses Verzeichnis: **C:\Programme\Gemeinsame Dateien\Microsoft Shared\Proof**
- Mit **Office2000** wird die Sache komplexer. Die Installation der *Office2000 Proofing Tools* dagegen verlief in meiner Umgebung einigermassen problemlos. Konkret hiess dies, ich musste nur einen technischen Artikel auf der *knowledge base* von Microsoft studieren und anschliessend einen kleinen Eingriff in der Registrierdatenbank vornehmen. Das ist jedoch guter Durchschnitt bei komplexeren Microsoft Produkten. Schliesslich haben diese ja auch ihren Preis. Sollten Sie weiterreichende Problem mit der Rechtschreibprüfung von bereits integrierten Sprachen, oder mit zusätzlichen Sprachen, welche Sie über die *Office2000 Proofing Tools* erworben haben, dann müssten Sie Sacheverhalt und Fehlermeldung bereitstellen. Greifen Sie in diesem Fall auch auf den technischen Support von Microsoft zurück.

Fremdsprachen welche die deutsche Ausgabe von Office 97 unterstützt

- Deutsch, inklusive Deutsch (Österreich), Deutsch (Schweiz), Deutsch (Luxemburg), Deutsch (Liechtenstein).
- Französisch, inklusive Französisch (Belgien), Französisch (Schweiz), Französisch (Kanada) und weitere Derivate, welche im französischsprechenden Teil Afrikas von Bedeutung sind.
- Englisch (US-amerikanisch und britisch) und viele andere englische Idiome.

Fremdsprachen welche die deutsche Ausgabe von Office2000 unterstützt?

- Deutsch, inklusive Deutsch (Österreich), Deutsch (Schweiz), Deutsch (Luxemburg), Deutsch (Liechtenstein).
- Französisch, inklusive Französisch (Belgien), Französisch (Schweiz), Französisch (Kanada) und weitere Derivate, welche im französischsprechenden Teil Afrikas von Bedeutung sind.
- Italienisch, inklusive Italienisch (Schweiz).
- Englisch (US-amerikanisch und britisch) und viele andere englische Idiome.

Wo kann ich die nötigen Dateien für die Rechtschreibeprüfung in weiteren Sprachen erwerben?

Office 97 - Proofing Tools von Alki Software

Was Office 97 betrifft vertreibet Microsoft keine ergänzenden Sprachtools, sondern verweist in dieser Sache auf die Firma **Alki Software Corporation**. Entwickelt wurden die Sprachtools schon zu früheren Zeiten von einer Firma mit dem Namen *SoftArt* welche heute *Novell* gehört. Gegenwärtig sind ca. dreissig Sprachen im Angebot. Wenn Sie sich einen Überblich über die zur Verfügung stehenden Sprachen verschaffen möchten, sollten Sie folgenden Link besuchen:

Der Preis für eine Sprachmodul beträgt z.Z. ca. € 90.00.

► **Rechtschreibtools für Fremdsprachen**

Office2000 Proofing Tools von Microsoft

Wie schon erwähnt, vertreibt Microsoft das **Office2000 Proofing Tools** Paket. Diese Software in Form einer CD-ROM erhalten Sie bei einem guten Softwarehändler für ca. € 190.00. Dieses Tool, welches ausschliesslich auf Office2000 angewendet werden kann, enthält rund 20 - 30 Sprachmoduls im *Bundle* (Gesamtpaket). Nach der Installation der Software, erscheint im Startmenü unter Microsoft Office Tools ein neuer Menüpunkt, welcher es Ihnen erlaubt, eine beliebige Anzahl von Sprachen zu aktivieren. Sie können dann einer Textpassage, einem Dokument oder einer Dokumentenvorlage diese Sprache über ***Extras - Sprache - Sprache bestimmen*** zuweisen. Welche Sprachen dieses Produkt unterstützt und weitere Einzelheiten entnehmen Sie bitte aus einem der Links, welcher unten aufgeführt ist.

Es ist auch möglich, ein Paket für eine einzelne Sprache zu erwerben. Der Preis dafür scheint aber im Vergleich zum Gesamtpaket nicht besonders attraktiv, liegt dieser doch bei ca. € 90.00.

Anmerkungen

- Unterschiedliche Sprachversionen von MS Office enthalten unterschiedliche Sprachtools. Wir Mitteleuropäer und Mitteleuropäerinnen fahren bezüglich der Anzahl der zur Verfügung stehenden Sprachen nicht schlecht. Dies ist u.a. der Vielsprachigkeit der Schweiz zu verdanken.
- Auf für Office 97 soll es möglich sein, ganze Sprachgruppen im Bundle (Gesamtpaket) zu erwerben. Der Preis soll dann vorteilhafter sein, als wenn Sie diese Sprachen einzeln erwerben. Allerdings liegen mir dazu zu wenig Informationen vor.

Links zu diesem Thema

Alki Software Corporation

Unter folgendem Link können Sie die Rechtschreibetools Online bestellen und bei geeignetem Zahlungsmittel gleich herunter laden. Der Preis für das Paket für eine Sprache liegt zur Zeit bei ca. € 90.

► **Alki Software Office Tools**

Microsoft

► **Microsoft Office2000 Proofing Tools**

Software Handel

► **HBSoft Preisliste für Office2000 Sofware**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/FSprache.htm

René Probst, Oktober 1999

Änderungen

- April 2000 - Office2000 Proofing Tools

Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?

◄ Zurück zum Hauptindex

Frage

> Ich sollte einen Brief in russischer Sprache und damit in kyrillischer Schrift verfassen. Wie komme ich nun zu meinen kyrillischen Zeichen?

Antwort

Um Ihrem Windows **baltisch**, **griechisch**, **kyrillisch** oder **türkisch** beizubringen sind **zwei** recht einfache **Schritte** nötige. Zuerst sollten Sie die Windows-Sprachunterstützung installieren. Informieren Sie sich, welche Sprachen bzw. Sprachgruppen Ihr Betriebssystem unterstützen kann. Angenommen, Sie setzen Windows ein, dann hängt die Verfügbarkeit der Sprachen neben der Software-Version auch noch von der Sprachversion ab. **Windows 95** Besitzer und Besitzerinnen, müssen wohl zuerst bei Microsoft die entsprechenden Komponenten herunterladen, wogegen **Windows2000** Benutzer im Trockenen sein dürften. **Windows 98** Besitzer sind auch nicht schlecht bedient. Meine deutschsprachige Version von Windows 98 z.B. unterstützt ohne zusätzliche Downloads folgende Sprachfamilien:

Sprachen bzw. Sprachfamilien, welche von Windows 98 unterstützt werden

- Baltisch
- Griechisch
- Kyrillisch
- Mitteleuropäisch
- Türkisch

Sprachunterstützung unter Windows (98) installieren

Um Ihren Rechner zu befähigen z. B. mit kyrillischen Zeichen umzugehen, führen Sie folgende Schritte aus. Für andere Sprachfamilien gehen Sie bitte analog vor.

1. Schmeissen Sie Ihre Windows 98 CD in das Laufwerk.
2. Rufen Sie unter Windows folgendes auf: Start - Einstellungen - Systemsteuerung - Software.
3. Wechseln Sie nun in den Reiter **Windows-Setup**.
4. Markieren Sie nun in der Liste den Eintrag **Sprachunterstützung**.
5. Klicken Sie auf die Schaltfläche ***Details....***
6. Wählen Sie nun eine der angebotenen Sprachfamilien, also z.B. *Kyrillisch*.
7. Schliessen Sie die Setup-Routine ab und starten Windows neu.

Damit sollte Ihre Windows-Installation grundsätzlich fähig sein, mit einer der genannten Schriften umzugehen. Damit Sie nun diese Zeichen über die Tastatur eingeben können, bedarf es noch eines zusätzlichen Tastaturtreibers. Auch dieser lässt sich in wenigen Schritten installieren. Windows unterstützt den parallelen Betrieb mehrerer Tastaturtreiber. Um z.B. den Tastaturtreiber für Russisch zu installieren gehen Sie wie folgt vor. Für andere Sprachen benutzen Sie bitte ein analoges Verfahren.

Zusätzlicher Tastaturtreiber installieren

1. Schmeissen Sie Ihre Windows 98 CD in das Laufwerk.
2. Rufen Sie unter Windows folgendes auf: Start - Einstellungen - Systemsteuerung - Tastatur.
3. Wechseln Sie nun in den Reiter **Sprache**.
4. Klicken Sie auf die Schaltfläche ***Hinzufügen...*** und wählen Sie dann die Sprache *Russisch*.
5. Schliessen Sie die Installation mit ***Ok*** ab.

Unmittelbar nach der Installation befindet sich unten rechts auf Ihrem Bildschirm (im sogenannten Systray) ein neues Symbol, welches es Ihnen erlaubt, fliegend zwischen den Tastaturtreibern zu wechseln. Alternativ können Sie auch mit der Tastenkombination **[Alt + Umschalt]** zwischen den Sprachen wechseln. Nun erscheinen also beim tippen einmal lateinische Buchstaben oder auf Wunsch eben kyrillische.

Sie sollten nun eigentlich nur noch um eine Tastatur besorgt sein, bei der die Tasten auch kyrillisch beschriftet sind. Grundsätzlich können Sie aber auch die auf Deutschland abgestimmte Tastatur verwenden und sich eine entsprechende Maske besorgen, falls sich so etwas aufreiben lässt.

Links zu diesem Thema

Eigene Links

► **Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen?**
► **Wo kann ich Wörterbücher für Fremdsprachen beziehen?**

Alfred Fleischli

► **Russisch und PC**

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Sprache.htm

René Probst, Mai 2000

Änderungen

- --- nil ---

Das Öffnen und Schliessen von Word-Dokumenten und anderen Office-Dateien, dauert immer länger

◄ Zurück zum Hauptindex

Symptome

> **Kurz nachdem ich Office 97 oder Office2000 installiert habe, gestaltete sich das Öffnen und Schliessen von Office-Dateien fix. Nun aber dauert es lange und ich habe den Eindruck, dass es immer schlimmer wird.**
>
> **Dieses Phänomen tritt übrigens auch dann auf, wenn Outlook gar nicht installiert ist, da es Office nicht hindert, mit dem Journaling fortzufahren (siehe unten). Falls Outlook installiert ist, können Sie das Journaling menügeführt ausschalten, andernfalls ist ein Eingriff in die Registrierdatenbank nötig.**

Ursache

Ein Feature namens Office-Journaling protokolliert standardmässig viele Vorgänge im Umgang mit Office-Dateien. Mit der Zeit wird die Datei, welche die Journalsätze enthält (es handelt sich dabei um die Datei mit dem Namen **offitems.log**), immer grösser. Als Folge davon wird der Zugriff auf diese Datei immer langsamer. Das Öffnen und Schliessen von Office-Dateien geht jetzt nur noch sehr schleppend von statten.

Abhilfe

- Löschen Sie die Datei **offitems.log**. Die Datei wird beim nächsten Start einer Office-Anwendung neu erstellt. (Hohe Priorität)
- Verhindern Sie, dass diese Datei wieder zu wuchern beginnt, in dem Sie das Journaling für die Office-Anwendungen ausschalten. Gehen Sie dabei wie folgt vor.
  - **Wenn Outlook 97/98 oder 2000 auf ihrem Rechner installiert ist:**
    1. Starten Sie Outlook
    2. Vom Hauptmenü wählen Sie ***Extras - Optionen***
    3. Wählen Sie den Reiter ***Einstellungen***
    4. Betätigen Sie die Schaltfläche ***Journaloptionen...***
    5. Vom Hauptmenü wählen Sie ***Extras - Optionen***
    6. Entfernen Sie alle Häkchen vor den Microsoft Office Anwendungen
    7. Schliessen Sie die Outlook-Anwendung
  - **Wenn Outlook 97/98 oder 2000 nicht auf ihrem Rechner installiert ist:**
    1. Wenn Sie ungeübt sind im Umgang mit der Registriereditor. lesen sie zuerst den Abschnitt:
       Der Umgang mit der Registrierdatenbank und dem Registriereditor
    2. Starten sie den **Registriereditor**
    3. Arbeiten Sie sich zum Schlüssel:
       `HKEY_CURRENT_USER\Software\Microsoft\Shared Tools\Outlook\Journaling\Microsoft Acces`
       vor und markieren Sie diesen Eintrag.
    4. Im Fensters auf der rechten Seite führen Sie nun einen Doppelklick auf den Eintrag **AutoJournaled** aus
    5. Im nun erscheinenden Dialog setzen Sie den Wert, welcher wenn das Journaling eingeschaltet ist *1* lautet auf **0**. Gegebenenfalls belassen Sie den Wert bei 0.
    6. Führen Sie die Schritte 3, 4 und 5 auch für die weiteren analogen Schlüssel aus. Die Namen dieser Schlüssel lauten:
       `HKEY_.....\Microsoft Excel, HKEY_.....\Microsoft PowerPoint`
       und `HKEY_.....\Microsoft Word.`

Anmerkungen

- Bei einem nicht vernetzten Rechner auf dem das Öffnen und Schliessen von Office-Dokumenten langsam geht, ist das Office-Journaling meistens **die** Bremse. Bei Rechner in einem Netzwerk kommen noch andere Überlegungen in Betracht.
- Nach einer Neuinstallation von Office oder der Installation eines der Office Service Releases, sollten Sie unbedingt kontrollieren, ob das Journaling durch die Installationsroutine nicht wieder eingeschaltet wurde. Gegebenenfalls schalten Sie es wieder aus.
- Wenn Sie anschauen möchten, was für Daten das Office-Journaling sammelt, können Sie dies selbstverständlich tun und anschliessend selbst entscheiden, ob diese Daten für Sie wichtig sind. Gehen Sie wie folgt vor:
  1. Starten Sie Outlook 97/98 oder Outlook2000.
  2. Wählen Sie in der **Outlook-Liste** das Symbol **Journal**. Falls dieses Symbol bei Ihnen nicht angezeigt wird, wechseln Sie zuerst zur Symbolleiste **Eigene Verknüpfungen**.

Links zu diesem Thema

Microsoft

Deutsch

► **OFF97: Öffnen und Schliessen von Programmen oder Dateien langsam**

Englisch

► **OL97: Outlook Starts Slowly with AutoJournal Feature**

► **WD97: Optimizing Microsoft Word 97**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Offlog.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie lässt sich die Standardschrift für neue Word-Dokumente nachhaltig ändern?

Wie lässt sich die automatische Silbentrennung nachhaltig aktivieren?

Wie lässt sich die Zoom-Einstellung des Dokumentenfenster nachhaltig ändern?

◄ Zurück zum Hauptindex

Symptome

> 1. Wenn ich ein neues Word-Dokument erstelle, erhalte ich immer die Schrift *Times New Roman* mit einem Schriftgrad von 10 Punkten. Ich möchte aber, wenn ich mit einem frischen Dokument beginne die Schrift Arial mit einem Schriftgrad von 12 Punkten.
>
> 2. Ich möchte, dass Word für mich die Silbentrennung automatisch ausführt. Für jedes Dokument, welches ich erstelle, muss ich dieses Feature aber wieder von neuem einschalten.
>
> 3. Ich möchte, dass mein Dokument mit 80% der Originalgrösse angezeigt wird. Aber jedes Mal, wenn ich Word neu starte, muss ich das von neuem einstellen.

Standardschrift nachhaltig ändern

Es gibt dafür zwei Lösungen, nämlich ein impliziter und ein expliziter Weg:

Standardschrift implizit setzen

1. Starten Sie Word.
2. Vom Word-Hauptmenü wählen Sie den Dialog ***Format - Zeichen***
3. Wählen Sie nun **Schriftart** und **Schriftgrad**
4. Betätigen Sie die Schaltfläche ***Standard...*** unten Links im Dialog
5. Bei der nun angezeigten Warnung bestätigen Sie Ihre Aktion.
6. Verlassen Sie Word. Falls Word Sie beim Verlassen anfragt, ob die Standardvorlage **Normal.dot** gespeichert werden soll, bejahen Sie dies.

Standardschrift explizit setzen

1. Lassen Sie nach der Standardvorlage **Normal.dot** suchen. Falls Sie mehrere Versionen dieser Datei auf Ihrem Rechner vorfinden, vergewissern Sie sich an Hand des Datum/Zeit-Eintrages im Windows-Explorer, dass Sie jene Standardvorlage gefunden haben, welche auch wirklich zum Tragen kommt.
2. **Öffnen** Sie diese Datei indem Sie über Rechtsklick - Kontextmenü den Punkt Öffnen wählen. Einen Doppelklick, ausgeführt auf eine Dokumentenvorlage, öffnet diese nicht, sondern erwirkt, dass ein neues Dokument auf Grund dieser Vorlage erstellt wird.
3. Sie können nun **Schriftart** und **Schriftgrad** wie gewohnt vorgeben.
4. Schreiben Sie nun einen kurzen belanglosen Text.
5. **Speichern** Sie die Vorlage nun.
6. Entfernen Sie nun das Geschriebene wieder.
7. Speichern Sie die Vorlage erneut und verlassen Sie Word.

Anmerkungen

- Dieser etwas umständliche Vorgang ist oft nötig, da aus Gründen der Performanz, die Standardvorlage Normal.dot gar nicht gespeichert wird, wenn nicht zusätzlich zur Änderung der Einstellungen, der Dokumententeil verändert wird.
- Dieses Vorgehen kann auch unter **PowerPoint** angewendet werden. Die Standardvorlage in PowerPoint heisst **Leere Präsentation.pot**.

Silbentrennung nachhaltig einschalten

1. Lassen Sie nach der Standardvorlage **Normal.dot** suchen. Falls Sie mehrere Versionen dieser Datei auf Ihrem Rechner vorfinden, vergewissern Sie sich an Hand des Datum/Zeit-Eintrages im Windows-Explorer, dass Sie jene Standardvorlage gefunden haben, welche auch wirklich zum Tragen kommt.
2. **Öffnen** Sie diese Datei indem Sie über Rechtsklick - Kontextmenü den Punkt Öffnen wählen. Einen Doppelklick, ausgeführt auf eine Dokumentenvorlage, öffnet diese nicht, sondern erwirkt, dass ein neues Dokument auf Grund dieser Vorlage erstellt wird.
3. Rufen Sie den Dialog Extras - Sprache - Silbentrennung... auf.
4. Schalten Sie die Option **Automatische Silbentrennung** und und bestätigen Sie Ihre Wahl mit **Ok**.
5. Schreiben Sie nun einen kurzen belanglosen Text.
6. **Speichern** Sie die Vorlage nun.
7. Entfernen Sie nun das Geschriebene wieder.
8. Speichern Sie die Vorlage erneut und verlassen Sie Word.

Anmerkung

- Dieser etwas umständliche Vorgang ist oft nötig, da aus Gründen der Performanz, die Standardvorlage Normal.dot gar nicht gespeichert wird, wenn nicht zusätzlich zur Änderung der Einstellungen, der Dokumententeil verändert wird.

Zoomfaktor nachhaltig ändern

1. Lassen Sie nach der Standardvorlage **Normal.dot** suchen. Falls Sie mehrere Versionen dieser Datei auf Ihrem Rechner vorfinden, vergewissern Sie sich an Hand des Datum/Zeit-Eintrages im Windows-Explorer, dass Sie jene Standardvorlage gefunden haben, welche auch wirklich zum Tragen kommt.
2. **Öffnen** Sie diese Datei indem Sie über Rechtsklick - Kontextmenü den Punkt Öffnen wählen. Einen Doppelklick, ausgeführt auf eine Dokumentenvorlage, öffnet diese nicht, sondern erwirkt, dass ein neues Dokument auf Grund dieser Vorlage erstellt wird.
3. Stellen Sie nun im Kombinationsfeld mit der Bezeichnung *Zoom* auf der Symbolleiste mit dem Namen *Standard* den gewünschten Wert ein. Sie können übrigens auch einen numerischen Wert rein schreiben, welcher nicht in der Liste erscheint.
4. Schreiben Sie nun einen kurzen belanglosen Text.
5. **Speichern** Sie die Vorlage nun.
6. Entfernen Sie nun das Geschriebene wieder.
7. Speichern Sie die Vorlage erneut und verlassen Sie Word.

Anmerkung

- Dieser etwas umständliche Vorgang ist oft nötig, da aus Gründen der Performanz, die Standardvorlage Normal.dot gar nicht gespeichert wird, wenn nicht zusätzlich zur Änderung der Einstellungen, der Dokumententeil verändert wird.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SSchrift.htm

René Probst, Oktober 1999

Änderungen

- 27. Februar 2001 - Gleiches Prozedere für Silbentrennung
- 27. Februar 2001 - Gleiches Prozedere für Zoomeinstellung

Wie kann ich eigene Untermenüs für AutoTexte erstellen?

◄ Zurück zum Hauptindex

Symptome

> **Wenn ich einen AutoText unter *Einfügen - AutoText* neu erstelle, landet dieser immer im Untermenü Standard. Dieses Untermenü schwillt aber bereits über. Ich möchte meine AutoTexte übersichtlicher organisieren.**

Lösung

Der Trick dabei ist, dass AutoTexte mit einer Formatvorlage verkoppelt sind. Um eine neue Kategorie zu eröffnen, müssen Sie zuerst eine Formatvorlage erstellen, welche exakt den selben Namen aufweisst, wie die gewünschte Kategorie im AutoText Untermenü.

Gesetzt den Fall, Sie möchten im AutoText-Menü eine Kategorie namens *Organische Chemie*, und dort die Einträge *Äthylalkohol*, *Äther* und *Essigsäure* erstellen, dann gehen Sie wie folgt vor:

1. Erstellen Sie ein neues Word-Dokument.
2. Wählen Sie vom Word-Hauptmenü den Dialog ***Format - Formatvorlagen... - Neu...***.
3. Im nun erscheinenden Dialog, tragen Sie im Feld **Name:** den Wert *Organische Chemie* ein.
4. Schliessen Sie diesen Dialog ab, in dem Sie nacheinander die Schaltflächen ***Ok*** und ***Zuweisen*** betätigen.
5. Im Kombinationsfeld **Formatvorlage** in der Symbolleiste Format sollte nun die Formatvorlage mit dem Namen *Organische Chemie* erscheinen.
6. Geben Sie nun folgenden Text ein: *Äthylalkohol* und anschliessend z.B. die Summenformel *C6H12O* für diesen chemischen Stoff.
7. Markieren Sie diesen Text.
8. Vom Word Hauptmenü wählen Sie nun den Dialog ***Einfügen - AutoText - Neu...***.
9. Den vorgeschlagenen Namen. nämlich *Äthylalkohol* übernehmen Sie am besten so und betätigen die Schaltfläche ***Ok***.
10. Nehmen Sie nun unter der selben Formatvorlage auch die übrigen chemischen Substanzen als AutoText auf.
11. Wenn Sie dies getan haben, können Sie auf Wunsch diese Formatvorlage verlustfrei löschen.
12. Löschen Sie nun den gesamten Inhalt des Dokumentes. Nun sollte wieder die Formatvorlage **Standard** aktiv sein. Ist dies nicht der Fall, sorgen Sie dafür.
13. Wenn Sie jetzt den Dialog ***Einfügen - AutoText*** wählen, sehen Sie Ihr neu erstelltes Untermenü mit dem Namen *Organische Chemie* und innerhalb dieses Menüs die Einträge, welche Sie soeben aufgenommen haben.

Anmerkungen

- Wenn die Einfügemarke an einem Ort steht, welche mit einer speziellen Formatvorlage formatiert wurde, erscheinen im AutoText-Menü nur diejenigen Einträge, welche unter dieser Formatvorlage aufgenommen wurden.
- Steht die Einfügemarke jedoch an einem Ort, welche mit der Formatvorlage **Standard** formatiert wurde, stehen alle AutoText-Einträge zur Verfügung.
- Standardmässig kennt Word bereits eine Anzahl von AutoText-Gruppen namentlich:
  - Anrede
  - Betreffzeile
  - Grußformel
  - Kopf- und Fußzeile
  - Versandanweisungen
- Auch hier gilt: diese Gruppen sind alle mit einer bestimmten Formatvorlage liiert.
- Sie können AutoText über den Word-Dialog ***Extras - Vorlagen und Add-Ins... - Organisieren...*** zwischen Vorlagen hin- und her kopieren.

Links zu diesem Thema

Microsoft

Deutsch

► **Word 97: AutoText-Gruppen in Word 8.0**

Englisch

► **WD97: Adding AutoText to the AutoText Menu**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/AutoText.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Positionsrahmen versus Textfelder

◄ Zurück zum Hauptindex

Problem

> In früheren Word-Versionen habe ich oft mit Positionsrahmen gearbeitet. Den Befehl Positionsrahmen einfügen kann ich jetzt aber nicht mehr finden.

Antwort

Positionsrahmen werde auch unter Word 97 und Word2000 weiterhin unterstützt, obwohl es inzwischen mit den **Textfeldern** einen veritablen Ersatz gibt. Textfelder bieten dem Benutzer oder der Benutzerin mehr Flexibilität. Den Befehl *Positionsrahmen einfügen* können Sie, wie die meisten alten Befehle, über den Word-Dialog **Extras - Anpassen** reaktivieren.

Befehl "Positionsrahmen einfügen" reaktivieren

Gehen Sie wie folgt vor:

1. Wählen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen...***.
2. Wechseln Sie zum Reiter ***Befehle***.
3. Markieren Sie in der linken Liste **Kategorien:** den Eintrag ***Alle Befehle***.
4. Suchen Sie in der rechten Liste **Befehle:** nach dem Eintrag ***EinfügenPosRahmen***.
5. Ziehen Sie diesen Befehl mit der Maus auf eine Symbolleiste Ihrer Wahl.
6. Verlassen Sie den Dialog mit der Schaltfläche ***Schliessen***

Funktionale Gegenüberstellung Positionsrahmen und Textfelder

- Textfelder bieten mehr Gestaltungsmöglichkeiten, was den Textfluss und den umgebenden Rahmen betrifft.
- Im Gegensatz zu Positionsrahmen, können Textfelder verknüpft werden. Damit wird es möglich, Text von einem Feld in das an fliessen zu lassen, wie wir dies von Desktop Publisher Programmen kennen.
- Während Positionsrahmen immer ein Rechteck darstellen, kennen Textfelder hier mehr Möglichkeiten. Nicht rechteckige Textfelder werden Legenden genannt. Um eine **Legende** einzubringen, lassen Sie sich die Symbolleiste **Zeichnen** anzeigen und wählen von dieser Leiste den Dialog ***AutoFormen - Legenden***.
- Im Gegensatz zu Positionsrahmen, lassen sich Textfelder gruppieren.
- Textfelder ermöglichen das Erstellen von Wasserzeichen
- Für Positionsrahmen spricht, dass diese Bestandteil einer Formatvorlage sein können und somit zur Einheitlichkeit des Dokumentes und zum Benutzerkomfort beitragen.. Dazu schreibt *Ralf Loerres folgendes:*

  > *Ich hatte das Problem, mit Rand-Anmerkungen arbeiten zu wollen, und da wollte ich nicht immer wieder dieselben Maßangaben eingeben. Wenn ich zurückging auf Word 95, konnte ich mich da bei Positionsrahmen bedienen, mit den Textfeldern von Word 97 geht das nicht. Ich weiß nicht, was man mit den Textboxen sonst noch so alles veranstalten kann, aber für mich war das sehr erleichternd, die Positionen von Hunderten von Positionsrahmen absatzbezogen per Formatvorlage zuweisen zu können.*

Probleme im Zusammenhang mit Textfeldern

Ungeschickt ist es, wenn Sie Beschriftungen in Textfeldern einfügen und anschliessend ein Abbildungsverzeichnis auf Grund dieser Beschriftungen generieren möchten. Da Word beim Erstellen des Verzeichnisses keine Textfelder durchsucht, wird auch nichts gefunden. Falls Sie also Beschriftungen überhaupt in einem Gefäss positionieren möchten, sind Sie mit Positionsrahmen besser bedient als mit Textfeldern.

Textfeldern nach Positionsrahmen konvertieren

Es ist möglich, Textfelder in Positionsrahmen zu konvertieren, nicht aber umgekehrt. Gehen Sie dabei wie folgt vor:

1. Markieren Sie das Textfeld am Rand.
2. Führen Sie einen ***Rechtsklick*** auf das Objekt aus.
3. Im erscheinenden Kontextmenü wählen Sie jetzt den Eintrag ***Textfeld formatieren***.
4. Wählen Sie nun den Reiter ***Textfeld***.
5. Nun betätigen Sie die Schaltfläche ***Konvertieren zu Positionsrahmen...***.
6. Word wird Sie jetzt warnen, dass gewisse Formatierungen verloren gehen können, Dieser Hinweis ist aber nicht auf den Text gemünzt. Ignorieren Sie also diese Warnung.

Links zu diesem Thema

Microsoft

► **Word 97: Positionsrahmen und Textfelder im Vergleich**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/PRahmen.htm

René Probst, Oktober 1999

Änderungen

- 29. Mai 2000 - Positionsrahmen können Bestandteil von Formatvorlagen sein

Das erste Wort nach einem Punkt wird immer mit einem Grossbuchstaben geschrieben

◄ Zurück zum Hauptindex

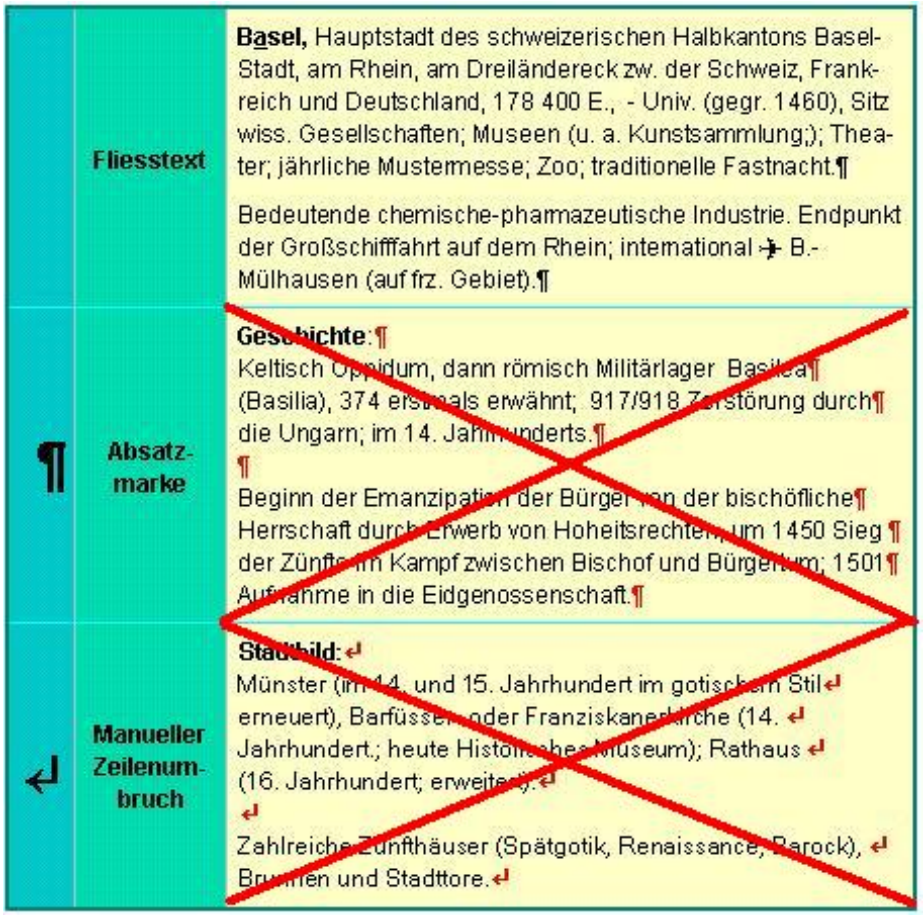
Symptome

> **Wenn Sie in Ihrem Dokument einen Punkt eingeben und anschliessend ein Wort welches mit einem Kleinbuchstaben beginnt eintippen, sorgt Word, - oft gegen Ihren Willen - dafür, dass dieser Buchstabe augenblicklich in einen Grossbuchstaben umgewandelt wird. Das selbe passiert auch, wenn Sie einen neuen Absatz beginnen, oder eine neue Zelle in einer Tabelle ausfüllen.**

Lösung

Fliesstext versus Zeilenumbrüche

Drücken Sie **nicht** die Eingabe-Taste am Ende einer Zeile. Im Gegensatz zu Schreibmaschinen brechen Textverarbeitungssysteme den Text am Ende der Zeile automatisch um. Dieses Verhalten wird **Fliesstext** genannt.



Bedeutung von Absätzen in einem Text

Mit dem Drücken der Eingabe-Taste tun Sie kund, dass Sie einen neuen Absatz wünschen. Einen neuen Absatz beginnt man, wenn ein neuer Gedankengang folgen soll. Eine neuer Absatz beginnt aber immer mit einem Grossbuchstaben. Damit ist das standardmässige Verhalten von Word folgerichtig und korrekt.

Manueller Zeilenwechsel

Wenn Sie aus irgend welchen Gründen doch frühzeitig eine neue Zeile beginnen möchten, also bevor der dynamische Zeilenumbruch des Textverarbeitungssystems zuschlägt, dann betätigen Sie die Tastenkombination **[Umschalt+Enter]**. Damit fügen Sie eine manuelle Zeilenumschaltung ein.

Nach einem manuellen Zeilenwechsel kann mit Kleinschreibung fortgefahren werden. Word greift in diesem Falle nicht ein.

Zusätzlichen Text vom bestehenden Text absetzen (Abstand schaffen)

Es ist im Zusammenhang mit Textverarbeitungssystemen eine schlechte Praxis unter (wiederholtem) Betätigen der Eingabe-Taste an eine Stelle weiter unten im Dokumenten zu gelangen, welche es gar noch nicht gibt (will heissen Abstand generieren). Das mag bei einem einseitigen Brief ja noch durchgehen, bei einem umfangreichen Textstück, kommt es aber einem Kunstfehler gleich, für welchen die Person, die das Dokument erbt, eigentlich Schadenersatz geltend machen müsste. Um Abstand zu schaffen, sollten Sie besser das Absatzformat anpassen. gehen Sie konkret wie folgt vor:

1. Rufen Sie vom Word-Hauptmenü den ***Dialog Format - Absatz...*** auf
2. Stellen Sie sicher, dass Sie sich im Reiter **Einzüge und Abstände** befinden
3. Stellen Sie im Textfeld mit der Bezeichnung **Abstand nach:** den gewünschten Abstand ein. Ein Punkt entspricht übrigens 1/72 eines Zolls (Inch, wobei 1 Inch = 2.54cm ist). Damit entsprechen 3 Punkte ungefähr einem Millimeter.

Absätze automatisiert mit zusätzlichem Abstand gegen einander abheben

Möchten Sie, dass Ihr Schriftstück lesbarer wird, gefällig aussieht und professionell weiterbearbeitet werden kann, dann sollten Sie sicherstellen, dass jeder Gedankengang einem Absatz entspricht, und sich dieser Absatz mit einem zusätzlichen Abstand (grösseren Durchschuss) von den umliegenden Absätzen optisch abhebt.

Drücken Sie also nicht nach jedem Augenblinzeln die Eingabe-Taste, sondern bei jedem neuen Atemzug.

Nutzen Sie dabei das Potenzial von Fliesstext. Verwenden Sie keine manuelle Zeilenwechsel (sie sind in den allen meisten Fällen völlig überflüssig). Betätigen Sie dann die Eingabe-Taste, wenn wirklich ein neuer Absatz beginnen soll und nicht wenn die Zeile zu Ende ist.

Vorgängig sollten Sie allerdings die Formatvorlage mit dem Namen **Standard** geschickt anpassen. Gehen Sie dabei wie folgt vor:

1. Rufen Sie vom Word-Hauptmenü den Dialog ***Format - Formatvorlage...*** auf.
2. Stellen Sie sicher, dass in der Liste der verfügbaren Formatvorlagen die Formatvorlage mit dem Namen **Standard** markiert ist.
3. Betätigen Sie nun die Schaltfläche **Bearbeiten...**.
4. Im folgenden Dialog aktivieren Sie als erstes die Option **Zur Vorlage hinzufügen**.
5. Klicken Sie nun auf die Schaltfläche ***Bearbeiten*** und wählen Sie im aufgeklappten Menü den Eintrag ***Absatz..***.
6. Stellen Sie sicher, dass Sie sich im Reiter **Einzüge und Abstände** befinden.
7. Im folgenden Dialog, im Abschnitt **Abstand**, stellen Sie die Werte für **Vor:** und **Nach:** auf 6 Punkte ein.

Umgehung

Sie können diesen Automatismus, welcher bewirkt. dass ein neuer Absatz zwingend mit einem Grossbuchstaben beginnt, auch ausschalten. Gehen Sie wie folgt vor:

1. Starten Sie Word.
2. Rufen Sie den Dialog ***Extras - AutoKorrektur*** auf.
3. Deaktivieren Sie die Option Jeden Satz mit eine, Grossbuchstaben beginnen.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Gross.htm

René Probst, Oktober 1999

Änderungen

- 29. November 2000 - Angereichert mit zusätzlichen nützlichen Informationen für Einstiger in die elektronische Textverarbeitung.

Ich möchte die Seitenzahl auf der ersten Seite nicht anzeigen.

Ich möchte für die ersten Seiten im Dokument römische Zahlen anzeigen.

◄ Zurück zum Hauptindex

Problemstellung

> **Ich habe ein umfangreiches Dokument. Der Verlag, dem ich das Skript abliefern muss, schreibt klare Richtlinien vor, bezüglich der Formatierung der Seitenzahlen. In meinem konkreten Fall heisst dies, dass auf der ersten Seite gar keine Seitenzahl erscheinen darf (diese Seite muss allerdings im folgenden mitgezählt werden), die Seite zwei bis ca. Seite fünf werden vom Inhaltsverzeichnis gebraucht und müssen mit römischen Zahlen nummeriert werden. Ab Seite sechs sollen dann allerdings arabische Zahlen zur Anwendung kommen.**

Lösung

MS Word bietet verschiedene Lösungsansätze für diese und ähnliche Vorgaben. Welche Lösung Sie letztlich verwenden sollten, hängt stark von den Ansprüchen ab. Die flexibelste Lösung basiert auf der Option, dass Sie Ihr Skript in mehrere Abschnitte unterteilen können. Dies hat den Vorteil, dass für jeden Abschnitt ein eigener Kopf- und Fusszeilenbereich zur Verfügung steht. Um genau zu sein, wenn Sie alle Einstellungen ausschöpfen (Erste Seite anders, Gerade und ungerade Seiten), stehen sogar für jeden Abschnitt drei Kopf- und Fussnotenbereiche zur Verfügung, nämlich für die erste Seite des Abschnittes, für ungerade Seiten des Abschnittes und für gerade Seiten des Abschnittes. Spätestens dann ist also volle Flexibilität gegeben. Sollten Sie aber all diese Optionen ausnützen, dann bedeutet dies für Sie einen grösseren Aufwand, den Sie müssen alle diese Abschnitte korrekt setzen und formatieren.

Die schnellste Lösungsvariante ist jene, welche es erlaubt, die erste Seite jedes Abschnittes bezüglich Kopf- und Fusszeile anders zu gestallten. Damit wird es auch möglich die Seitenzahlen auf der ersten Seite zu unterdrücken. Falls Ihr Dokument auf einem einzigen Abschnitt beruht (und dies ist der Fall, solange Sie keine Abschnittswechsel im Dokument setzen), sind Sie sehr schnell am Ziel. Allerdings können Sie damit nur die erste Seite anders gestalten

Seitenzahlen mit römischen Zahlen anzeigen zu lassen, ist kein grosses Kunststück mit Word. Auch diese Schikane Ihres Verlages, werden Sie mit der folgender Anweisung problemlos bewältigen.

Erste Seite anders

Soll sich die erste Seite bezüglich den anderen Seiten unterscheiden, dann gehen Sie wie folgt vor:

1. Vom Word-Hauptmenü wählen Sie den Dialog ***Datei - Seite einrichten...***
2. Wechseln Sie zum Reiter ***Seitenlayout.***
3. Im Bereich **Kopf- und Fußzeilen** aktivieren Sie die Option ***Erste Seite anders.***
4. Schliessen Sie den Dialog mit ***Ok*** ab.

Damit stellt Word Ihnen einen eigenen Kopf- und Fusszeilenbereich für die erste Seite zu Verfügung. Stellen Sie nun am besten sicher, dass sich die Einfügemarke am Anfang des Dokumentes befindet.

Möchten Sie nun erwirken, dass die Seitenzahl auf der ersten Seite nicht angezeigt wird, wechseln Sie als erstes in die Kopf- und Fusszeilenansicht. Angenommen die Seitenzahlen befinden sich grundsätzlich in der Fusszeile wechseln Sie mit der kontextbezogenen Symbolleiste zur Fusszeile. Der entsprechende Schalter heisst *Zwischen Kopf- und Fusszeile wechseln*..

Dieser Fusszeilenbereich sollte nun neu die Bezeichnung **Erste Fußzeile** aufweisen. Ist dies nicht der Fall, dann befinden Sie sich wahrscheinlich nicht auf der ersten Seite. Als nächstes markieren Sie nun den Bereich in dem die Seitenzahl angezeigt wird. Es handelt sich dabei um eine Feldfunktion. Die Hintergrundsfarbe des Bereiches sollte sich von hellgrau nach schwarz ändern. Sie sollten nun, da Sie ja auf der ersten Seite keine Seitenzahl möchten, diesen dunkel markierten Bereich löschen.

Wenn Sie die Seitenzahl gelöscht haben, können Sie sich mit der Schaltfläche ***Nächste anzeigen*** (diese Schaltfläche finden Sie auf der kontextbezogenen Symbolleiste) zum nächsten Fusszeilenbereich vorarbeiten. Die Bezeichnung des Bereiches sollte nun nicht mehr *Erste Fußzeile* lauten sondern nur noch **Fußzeile**. Hier fügen Sie nun immer noch über die kontextbezogene Symbolleiste die Seitenzahlfunktion ein. Bei Bedarf können Sie das Aussehen der Seitenzahlanzeige noch nachformatieren. Auch die Position der Seitenzahl lässt sich beeinflussen, wohl am effizientesten, wenn Sie Tabulatoren verwenden. Normalerweise verfügt die Formatvorlage für die Fusszeile über einen rechtsbündigen Tabulator am rechten Rand.

Verlassen Sie nun die Kopf- und Fusszeilenansicht und überprüfen Sie Ihr Werk in der Seitenlayout-Ansicht.

Die ersten fünf Seiten sollen mit römischen Zahlen ausgestattet werden

Zusammenfassend sollten Sie folgendes tun:

- Teilen Sie Ihr Dokument in Abschnitte ein.
- Entkoppeln Sie Kopf- bzw. Fusszeilen der einzelnen Abschnitte.

Hier nun eine Schritt für Schritt Anweisung für die geschilderte Problemstellung:

1. Setzen Sie die Einfügemarke ganz ans Ende der fünften Seite.
2. Vom Word Hauptmenü wählen Sie ***Einfügen - Manueller Wechsel... - Abschnittswechsel - Nächste Seite***. Damit besteht Ihr Dokument nun aus zwei Abschnitten.
3. Gehen Sie ganz an den Anfang des Dokumentes.
4. Wechseln Sie in die Kopf- und Fusszeilenansicht.
5. Wechseln Sie allenfalls mit der entsprechenden Schaltfläche auf der kontextbezogenen Symbolleiste von der Kopfzeile in die Fusszeile.
6. Der Fusszeilenbereich sollte nun die Bezeichnung **Fußzeile Abschnitt 1** haben.
7. Fügen Sie nun die Seitenzahlfunktion über die entsprechende Schaltfläche auf der kontextbezogenen Symbolleiste an der gewünschten Stelle in der Fusszeile ein.
8. Wählen Sie immer noch auf dieser Symbolleiste den Schalter ***Seitenzahlen formatieren***.
9. Im nun erscheinenden Dialog wählen Sie im Kombinationsfeld ***Zahlenformat*** entweder die Zahlenreihe mit klein- bzw. gross geschriebenen römischen Zahlen.
10. Schliessen Sie diesen Dialog mit ***Ok*** ab. Verweilen Sie aber in der Kopf- bzw. Fusszeilenansicht.
11. Betätigen Sie nun auf der Symbolleiste den Schalter ***Nächste anzeigen.***
12. Es wird nun der Bereich mit der Bezeichnung **Fußzeile Abschnitt 2** angezeigt.
13. Vergewissern Sie sich, dass dort arabische Seitenzahlen erscheinen. Allenfalls schalten Sie das Zahlenformat um.
14. Verlassen Sie nun die Kopf- und Fusszeilenansicht und überprüfen Sie Ihr Werk in der Seitenlayout-Ansicht.

Anmerkungen

- Die Funktion ***Einfügen - Seitenzahlen*** vom Word-Hauptmenü sollten Sie für komplexere Aufgabenstellungen besser nicht verwenden. Sie bietet dazu nicht die nötige Flexibilität.
- Über den Dialog Seitenzahl formatieren, können Sie auch angeben, wenn die Nummerierung eines Abschnittes **nicht bei eins beginnen** sollte.
- Bei der Seitennummerierung handelt es sich um eine Feldfunktion. Sie können deshalb auch Seitenzahlen über den Word-Dialog ***Einfügen - Feld... -*** Feldname ***Seite*** einfügen, falls dies auf Grund der Aufgabenstellung sinnvoll erscheint. In **Word2000** heisst die selbe Feldfunktion nicht *Seite* sondern **Page**.
- Diese Feldfunktion eignet sich auch für kleine arithmetische Turnübungen. Die Feldfunktion **{ = { Seite } +2 }** z. B. liefert die tatsächlich Seitenzahl, zählt aber noch zwei dazu. Die geschweiften Klammern müssen sie selbstverständlich mit der Tastenkombination **Strg + F9** generieren, sollte das Experiment gelingen.
- Auch eine **WENN-Abfrage** ist statthaft. Das könnte dann etwa wie folgt aussehen:
  { WENN { Seite } < { ANZSEITEN } "Fortsetzung folgt" "Ende der Durchsage." }
  Word 97

  { IF { Page } < { NumPages } "Fortsetzung folgt..." "Ende der Durchsage." }
  Word2000

  Falls Ihnen dieses Experiment nicht auf Anhieb gelingt, beachten Sie bitte folgendes:
  - Sowohl die Seitenfunktion sowie die Anzahlseitenfunktion, müssen sich im gleichen Dokumententeil befinden, also z. B. in der Fusszeile.
  - Nach jeder Korrektur der Feldfunktion sollten Sie die Felder aktualisieren.
  - Seitenzahlen aktualisieren sich in Word nicht zuverlässig. Sie sollten deshalb **mehrmals** zwischen Seitenansicht und Normalansicht hin- und her schalten oder es gleich direkt mit einem Ausdruck versuchen.

Links zu diesem Thema

Eigene Links

► **Wie drucke ich in meinem Dokument Seitenzahlen nach der DIN?**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SZahlen2.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie drucke ich in meinem Dokument Seitenzahlen nach der DIN?

◄ Zurück zum Hauptindex

Problemstellung

> Eine neue Norm schreibt vor, dass Dokumente nach einem neuen Prinzip durchnummeriert werden sollen. In der Kopfzeile soll die aktuelle Seitenzahl erscheinen und in der Fusszeile soll bei mehrseitigen Dokumenten, die Seitenzahl der nächsten Seite referenziert sein, es sei denn, es handle sich um die letzte Seite. Wie kriege ich diese Sache mit Word geritzt?

Lösung

Wechseln Sie in die Kopfzeile und geben Sie dort, wie Sie dies gewohnt sind, das Feld für die Seitenzahl ein, z. B. über die Schaltfläche **Seitenzahl einfügen** in der kontextbezogenen Symbolleiste.

Für die Fusszeile ist eine etwas kniffligere Operation angesagt, aber auch dies lässt sich lösen und zwar mit einer Feldfunktion. Diese Feldfunktion lautet wie folgt:

Word 97

```
{ Wenn { AnzSeiten } > 1 { Wenn { Seite } < { AnzSeiten } { =
{ Seite } + 1 } "" } "" }
```

Word2000

```
{ IF { NumPages } > 1 { IF { Page } < { NumPages } { = { Page
} + 1 } "" } "" }
```

So bringen Sie einen Feldcode in Ihr Dokument ein

1. Wechseln Sie mit der Tastenkombination **[Alt + F9]** in die Feldcodeansicht. Dabei handelt es sich um einen sogenannten **Toggle-** (Ein- Ausschalter).
2. Setzen Sie die Einfügemarke an die Stelle an welcher der Feldcode bzw. dessen Resultat erscheinen soll.
3. Die geschweiften Klammern dürfen Sie nicht auf konventionelle Art über die Tastatur eingeben. Verwenden Sie dazu die Tastenkombination **[Strg + F9]** jedes mal, wie Sie diese Tastenkombination auslösen, entsteht ein Klammer-**paar**. In solch generierte Klammern, sollten Sie den Code nun eingeben.

Anmerkung

! Nach jeder Korrektur des Codes, sollten Sie den Code aktualisieren, bevor Sie das Resultat beurteilen. Sie erreichen dies, indem Sie den Feldcode markieren und die Taste [F9] betätigen. Der Feldschatten sollte beim markieren dunkel, sprich schwarz werden.

VBA-Code

Für diese etwas drollige Norm, hier eine Makrolösung. Der VBA-Code setzt den hier abgehandelten Feldcode **linksbündig**. Sie können die Ausrichtung der Seitenzahlen aber im Kopf des Codes auf sehr einfache Wiese anpassen. Setzen Sie die Konstante mit dem Namen *Ausrichtung* auf **1** wenn die Seitenzahlen **rechtsbündig** erscheinen sollen bzw. auf **2** wenn die Seitenzahlen **zentriert** erscheinen sollen

```
Private Const Ausrichtung = 2 ' 0: Linksbündig
                              ' 1: Rechtsbündig
                              ' 2: Zentriert

Private Const FB = False      ' True oder False (Format behalten)

Sub SeitenZahlenNachDINEinfügen()
  Dim Fuss As HeaderFooter, oRange As Range, Feld As Field
  Set Fuss = ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary)
  Fuss.Range.Delete
  With ActiveDocument.Sections(1).PageSetup
    rechts = .PageWidth - .LeftMargin - .RightMargin
    zentriert = rechts / 2
  End With
  With Fuss.Range.ParagraphFormat
    .TabStops.ClearAll
    Select Case Ausrichtung
      Case 0
      Case 1
        .TabStops.Add Position:=rechts, Alignment:=wdAlignTabRight
        Fuss.Range.InsertAfter vbTab
      Case 2
        .TabStops.Add Position:=zentriert, Alignment:=wdAlignTabCenter
        Fuss.Range.InsertAfter vbTab
    End Select
  End With
  Set oRange = Fuss.Range
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  Set Feld = Fuss.Range.Fields.Add(Range:=oRange, Type:=wdFieldIf, _
     PreserveFormatting:=FB)
  oRange.SetRange Start:=Feld.Code.End - 1, End:=Feld.Code.End - 1
  Fuss.Range.Fields.Add Range:=oRange, Type:=wdFieldNumPages, PreserveFormatting:=FB
  oRange.SetRange Start:=Feld.Code.End - 1, End:=Feld.Code.End - 1
  oRange.InsertAfter " > 1 "
  Set oRange = Fuss.Range.Fields(1).Code
  oRange.SetRange Start:=oRange.End - 1, End:=oRange.End - 1
  Set Feld = oRange.Fields.Add(Range:=oRange, Type:=wdFieldIf, _
     PreserveFormatting:=FB)
  oRange.SetRange Start:=Feld.Code.End - 1, End:=Feld.Code.End - 1
  Set Feld = Fuss.Range.Fields.Add(Range:=oRange, Type:=wdFieldPage, _
     PreserveFormatting:=FB)
  oRange.SetRange Start:=Feld.Code.End + 3, End:=Feld.Code.End + 3
  oRange.InsertAfter " < "
  oRange.SetRange Start:=Feld.Code.End + 6, End:=Feld.Code.End + 6
```

```
  Set Feld = oRange.Fields.Add(Range:=oRange, Type:=wdFieldNumPages, _
     PreserveFormatting:=FB)
  oRange.SetRange Start:=Feld.Code.End + 3, End:=Feld.Code.End + 3
  oRange.Fields.Add Range:=oRange, Type:=wdFieldEmpty, Text:="= ", _
     PreserveFormatting:=FB
  oRange.SetRange Start:=Feld.Code.End + 6, End:=Feld.Code.End + 6
  Set Feld = Fuss.Range.Fields.Add(Range:=oRange, Type:=wdFieldPage, _
     PreserveFormatting:=FB)
  oRange.SetRange Start:=Feld.Code.End + 3, End:=Feld.Code.End + 3
  oRange.InsertAfter " + 1 "
  Set oRange = Fuss.Range.Fields(1).Code
  oRange.SetRange Start:=oRange.End - 1, End:=oRange.End - 1
  oRange.InsertAfter Chr(34) & Chr(34)
  Set oRange = Fuss.Range.Fields(3).Code
  oRange.SetRange Start:=oRange.End - 1, End:=oRange.End - 1
  oRange.InsertAfter Chr(34) & Chr(34)
End Sub
```

(Einfügen der Seitenzahlen in die Fusszeile nach Deutscher Industrie Norm.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SZahlen3.htm

René Probst, Oktober 1999

Änderungen

- April 2000 - VBA-Code flexibler geschrieben (rechtsbündig, zentriert)
- Februar 2001 - VBA-Code optimiert (Arbeiten mit Range-Objekten / Kein Wechsel in die Kopf-/Fusszeilenansicht.

Wo werden Bausteine wie Formatvorlagen, AutoTexte, Symbolleisten, AutoKorrektur- und Benutzerwörterbucheinträge gespeichert?

◄ Zurück zum Hauptindex

Diese Seite

▼ Formatvorlagen, AutoTexte und Symbolleisten
▼ AutoKorrektur-Einträge
▼ Sichern, Wiederherstellen und Portieren von Autokorrektur-Einträgen
▼ Konvertieren der AutoKorrektur-Einträge von Word 6.0 nach Word 95, 97 oder 2000
▼ Benutzerwörterbücher
▼ Links zu diesem Thema

Formatvorlagen, AutoTexte und Symbolleisten

▲ Zurück zum Anfang

Alle diese drei Komponenten werden in einer Dokumentenvorlage gespeichert. Wenn Sie keine speziellen Dokumentenvorlagen auf Ihrem Rechner eingerichtet haben, dann befinden sich diese Bausteine in der Standarddokumentenvorlage, welche bei jeder Word-Installation **Normal.dot** heisst und immer beim Aufstarten von Word geladen wird. Es ist auch möglich, diese Bausteine in einem bestimmten Dokument zu speichern, allerdings ist dies in der Praxis weniger üblich.

Diese Bausteine können auf bequeme Art und Weise administriert werden, das heisst Sie können diese Komponenten zwischen Vorlagen verschieben oder kopieren. Die Bausteine können auch gelöscht oder unbenannt werden. Gehen Sie wie folgt vor:

1. Rufen Sie vom Word-Hauptmenü den Dialog Extras - Vorlagen und Add-Ins... - Organisieren... auf.
2. Dieser Dialog setzt sich aus vier Registern zusammen, namentlich: *Formatvorlagen* - *AutoTexte* - *Symbolleisten* und *Makros*.
3. Über die Schaltfläche ***Datei schliessen*** bzw. ***Datei öffnen*** können Sie sowohl die Quellvorlage wie auch die Zielvorlage vorgeben.
4. Dank den Schaltflächen ***Kopieren***, ***Löschen*** und ***Umbenennen...*** können die genannten Bausteine administriert werden.

Anmerkung zu Symbolleisten

Bezüglich dem Administrieren von Symbolleisten gibt es etwas Wichtiges zu bedenken. Im Gegensatz zu benutzerdefinierten Symbolleisten können word-eigene Symbolleisten nicht wie beschrieben organisiert werden. Sollten Sie also integrierte Symbolleisten nach Ihren Wünschen angepasst haben, bleibt Ihnen nur der Umweg über eine temporär erstelle Symbolleiste. Gehen Sie in diesem Falle wie folgt vor:

1. **Öffnen** Sie die alte Vorlage.
2. Wählen Sie aus dem Word-Hauptmenü den Dialog ***Extras - Anpassen...***.
3. Wechseln Sie zum Reiter ***Symbolleisten***.
4. Betätigen Sie die Schaltfläche ***Neu...***.
5. Erstellen Sie eine neue Symbolleiste mit einem belanglosen Namen.
6. Halten Sie die Taste **[Strg]** gedrückt und ziehen Sie das fragliche Element auf die neue Symbolleiste.
7. Verlassen Sie den Dialog mit der Schaltfläche ***Schliessen***.
8. Damit die temporäre Symbolleiste gesichert wird, speichern Sie nun die alte Vorlage.
9. **Öffnen** Sie nun die neue Vorlage.
10. Wählen Sie aus dem Word-Hauptmenü den Dialog Extras - Vorlagen und Add-Ins... - Organisieren....
11. Wechseln Sie zum Reiter ***Symbolleisten***.
12. Betätigen Sie auf der rechten Seite des Dialoges die Schaltfläche ***Datei Schliessen.***.
13. Betätigen Sie die selbe Schaltfläche, welche nun ***Datei Öffnen*** heisst nochmals.
14. Wählen Sie in der Liste den Name der alten Vorlage aus.
15. Markieren Sie die temporär erstellte Symbolleiste und betätigen Sie die Schaltfläche ***Kopieren***.
16. Verlassen Sie den Dialog mit der Schaltfläche ***Schliessen***.
17. Wählen Sie aus dem Word-Hauptmenü den Dialog ***Extras - Anpassen...***.
18. Wechseln Sie zum Reiter ***Symbolleisten***.
19. Suchen Sie in der Liste nach der temporär erstellten Symbolleiste und aktivieren Sie das Häkchen.
20. Ziehen Sie die Elemente auf der temporären Symbolleiste auf eine Symbolleiste Ihrer Wahl.
21. Entfernen Sie nach Möglichkeit alle redundanten Befehle von den Symbolleisten.
22. Verlassen Sie den Dialog mit der Schaltfläche ***Schliessen*** und speichern Sie die Vorlage.

Um diese längere Übung zu umgehen, legen Sie nach Möglichkeit Schaltflächen, welche Sie verändert haben bzw. erstellt haben auch in benutzerdefinierten Symbolleisten an.

AutoKorrektur-Einträge

▲ Zurück zum Anfang

Die meisten AutoKorrektur-Einträge werden in einer externen Datei aufbewahrt. Diese Datei weist die Dateierweiterung **.acl** auf und befindet sich bei einer typischen Installation im Windows-Verzeichnis. In der Regel kennt Ihr System so viele acl-Dateien, wie Benutzer auf Ihrem Rechner eingerichtet sind. Wenn Sie nicht sicher sind, welche AutoKorrektur-Datei aktiv ist, können Sie zu Testzwecken einen neuen AutoKorrektur-Eintrag erstellen. Gleich anschliessend lassen Sie nach den Dateien mit der Erweiterung "acl" suchen und vergleichen im Windows-Explorer den Zeiteintrag.

Nicht alle AutoKorrektur-Einträge werden in dieser *acl-Datei* abgelegt. Grafische AutoKorrektur-Einträge, wie z. B. Logos, Einträge welche Feldcode enthalten oder Einträge, welche mit der Symbolschrift Windings formatiert wurden, befinden sich in der Standardvorlage **Normal.dot**. Als Konsequenz davon, stehen diese speziellen Einträge auch nur in Word zur Verfügung. Andere Office-Anwendungen können davon also nicht profitieren.

Für **Office2000** gilt zusätzlich, dass diese *acl-Dateien* **sprachsensitiv** sind. Letzteres macht ja auch Sinn, denn eine Buchstabenkombination, welche in der Sprache A ein eindeutiger Vertipper darstellt, kann in der Sprache B ein korrektes und gebräuchliches Wort sein

Sichern, Wiederherstellen und Portieren von Autokorrektur-Einträgen

▲ Zurück zum Anfang

Microsoft liefert eine Dokumentenvorlage namens **Makro80.dot** aus. Darin enthalten ist ein Makro welches es ermöglicht, Autokorrektur-Einträge zu sichern und zwar unabhängig davon, ob diese nun in der *acl-Datei* oder in der Normal.dot gespeichert sind. Mit dem selben Werkzeug können diese Einträge auch wieder zurückgespielt werden oder auf zu einem anderen Rechner getragen werden. Gehen Sie wie folgt vor:

1. Lassen Sie auf Ihrem Rechner nach einer Datei namens **Makro80.dot** suchen. Sollte sich diese Dokumentenvorlage noch nicht auf Ihrem Rechner befinden, schmeissen Sie Ihre **Office 97**-CD in das Laufwerk und lassen dort nach dieser Datei suchen. Am Besten kopieren Sie anschliessend diese Dokumentenvorlage auf Ihre Festplatte.
2. Benutzer von **Office2000** finden das Gegenstück, nämlich die Datei mit dem Namen **Macros9.dot** vielleicht auf Ihrer CD. Zur Zeit als wir Office2000 erworben haben, war diese Dokumentenvorlage jedoch noch nicht auf der CD-ROM enthalten. Wo Sie diese Dokumentenvorlage herunterladen können, wird am Ende des Skriptes erwähnt.
3. Führen Sie einen Doppelklick auf die genannte Dokumentenvorlage aus.
4. Wählen Sie im Wordhauptmenü den Punkt ***Extras - Makro - Makros...*** und führen Sie das Makro mit dem Namen **AutoCorrectUtil** aus.
5. Der Rest des Vorgangs ist dialoggeführt und eigentlich selbsterklärend.

Sprachsensitive Autokorrektur-Einträge in Word2000 sichern und wiederherstellen

Wie schon angedeutet sind die AutoKorrektur-Einträge in Word2000 sprachsensitive. Möchten Sie AutoKorrektur-Einträge einer anderen Sprache als der geraden aktiven sichern, müssen Sie diese Sprache zuerst zum Standard küren. Gehen Sie wie folgt vor:

1. Starten Sie Word.
2. Wählen Sie im Word-Hauptmenü den Dialog ***Extras - Sprache - Sprache bestimmen...***.
3. Wählen Sie in der Liste die Sprache aus, deren AutoKorrektur-Einträge Sie ändern möchten. Es muss sich dabei um eine Sprache handeln, welche im Lieferungsumfang Ihrer Office2000-Ausgabe enthalten ist.
4. Betätigen Sie die Schaltfläche ***Standard...***.
5. Bei der nun folgenden Sicherheitsabfrage, bestätigen Sie Ihre Aktion nochmals.
6. **Wichtig:** Schliessen Sie Word nun.
7. Falls beim Schliessen eine Anfrage erscheint, ob die Normal.dot gespeichert werden soll, bejahen Sie dies.
8. Öffnen Sie nun die Dokumentenvorlage mit dem Namen **Macros9.dot**, welche Sie vom Microsoft Server herunter laden können.
9. Führen Sie einen Doppelklick auf die genannte Dokumentenvorlage aus.
10. Wählen Sie im Wordhauptmenü den Punkt ***Extras - Makro - Makros...*** und führen Sie das Makro mit dem Namen **AutoCorrectUtil** aus.
11. Der Rest des Vorgangs ist dialoggeführt und eigentlich selbsterklärend.
12. Bei Bedarf können Sie die Standardsprache nun wieder zurücksetzen.

Konvertieren der AutoKorrektur-Einträge von Word 6.0 nach Word 95, 97 oder 2000

▲ Zurück zum Anfang

In der Word-Version 6.0 waren die AutoKorrektur-Einträge noch auf eine andere Weise organisiert. Alles Nötige war damals in einer Dokumentenvorlage vereint. Öffnen Sie eine Dokumentenvorlage aus dieser Zeit z. B. unter Word 97, wird die Vorlage zwar konvertiert, die AutoKorrektur-Einträge wandern aber nicht mit. Sie sind zwar immer noch in der Vorlage enthalten, sie werden aber nicht in der entsprechenden Liste angezeigt und ein Zugriff darauf ist auch nicht möglich.

Der Theorie nach, findet nach dem ersten Start von Word nach erfolgreicher Installation von Office 95/97 eine automatische Konvertierung der AutoKorrektur-Einträge aus der alten "Normal.dot" statt. In Anbetracht der zahlreichen Anfragen in dieser Sache, scheint es damit aber in der Praxis zu harzen. Sollte dieser Automatismus also auch bei Ihnen versagen, können Sie wie folgt vorgehen:

1. Benennen Sie die "acl-Datei", welche sich üblicherweise im Windows-Verzeichnis befindet und mit der ID beginnt unter welcher Sie sich bei Windows anmelden, um.
2. Nennen Sie die Standarddokumentenvorlage "Normal.dot" um.
3. Löschen Sie folgende zwei Schlüssel in der Registrierdatenbank:
   - `HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data`
   - `HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Common\AutoCorrect`
4. Starten Sie Word. Die Konvertierung sollte nun automatisch stattfinden.

Falls Sie die AutoKorrektur-Einträge von Word 95 (alias Word 7.0) nach Word 97 übernehmen müssen, ist das Vorgehen analog. Schauen Sie sich in diesem Falle den unten eingeblendeten Link an.

Benutzerwörterbücher

▲ Zurück zum Anfang

Wörter, welche Sie der Rechtschreibprüfung hinzufügen, werden standardmässig in einer Datei mit dem Namen **Benutzer.dic** abgelegt. Es handelt sich dabei um eine einfache Textdatei, wobei die einzelnen Wörter mit einem Zeilenumbruch gegeneinander abgegrenzt sind. Sie können diese Datei, welche sie im Unterverzeichnis **Proof** finden, problemlos von einem zum anderen Rechner tragen und allenfalls z.B. mit dem kleinen Windows-Programm **Notepad.exe** editieren. Verwenden Sie keine Ausdrücke, welche Leerschläge beinhalten. Es passiert zwar nichts, aber Word kann aus solchen Begriffen nichts machen. Word stört sich nicht an allfälligen Dubletten, auf eine alphabetische Sortierreihenfolge der Wörter brauchen Sie nicht zu achten. Word sorgt von sich aus für die alphabetische Sortierreihenfolge.

Mit unter finden Sie in der Wörterbuchdatei auf der ersten Zeile eine Kennung, welche etwa wie folgt aussieht: **#LID 1031**, wobei "LID" für *Language ID* steht. Auch nach Aussage von Microsoft darf man diese Kennung löschen, oder man kann sie auch stehen lassen. Es handelt sich offenbar um eine redundante Information, welche beim Fehlen die Funktionalität nicht negativ beeinflusst.

Neben dem Standardbenutzerwörterbuch mit dem Namen *Benutzer.dic* können Sie auch noch weitere gleichgestaltete Wörterbücher verwenden. Sie können Word bis **maximal 10** solcher Wörterbücher unterschieben, die z. B. Fachwörter nach Thematik getrennt enthalten. Ein einzelnes Wörterbuch kann bis zu **5000 Wörter** aufnehmen. Allerdings ist bei einer **Dateigrösse von 64KB** Schluss unabhängig davon, ob diese bereits 5000 Wörter enthält. Nach erreichen einer dieser Limiten, passiert nichts mehr, falls Sie weitere Wörter hinzufügen möchten. Es wird leider keine Fehlermeldung ausgegeben; im übrigen arbeitet Word aber ohne Probleme weiter. Die folgenden Anweisung zeigt, wie Sie ein spezielles Wörterbuch mit Word verbinden können.

1. Schreiben Sie ein falsch geschriebenes Wort.
2. Starten Sie die Rechtschreibprüfung im Vordergrund über den Dialog ***Extras - Rechtschreibung und Grammatik...***.
3. Betätigen Sie in diesem Dialog die Schaltfläche ***Optionen...***.
4. Im nun folgenden Dialog wählen Sie die Schaltfläche ***Wörterbücher***.
5. Wählen Sie nun die Schaltfläche ***Hinzufügen...*** oder ***Neu...***, je nach dem ob das Wörterbuch bereits besteht.
6. Schliessen Sie diesen Dialog mit ***Ok - Ok - Abbrechen*** ab.

Links zu diesem Thema

▲ Zurück zum Anfang

Microsoft

Deutsch

► Verschieben von AutoKorrektur-Einträgen zwischen Computern
► Microsoft Word2000-Zusatzmakros: Macros9.dot
► AutoKorrektur-Einträge von Word95 nach Word 97 übernehmen

Englisch

► Where AutoCorrect Entries and Settings Are Stored
► WD: How to Convert Word 6.x AutoCorrect Entries

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Baustein.htm

René Probst, Oktober 1999

Änderungen

- 14. November 1999 - Benutzerwörterbuch aufgenommen

Wo speichert MS Word seine Einstellungen (Umgebungsoptionen)?

◄ Zurück zum Hauptindex

Diese Seite

▼ Frage
▼ Voranbemerkungen
▼ Formatvorlagen als Trägerinnen von Einstellungen
▼ Die Bedeutung der Standardvorlage mit dem Namen «Normal.dot»
▼ Anwendungsweite Einstellungen in der Registrierdatenbank
▼ Word-Bausteine
▼ Benutzerdefinierte Etiketten
▼ Fazit
▼ Installationswerte wieder herstellen (Alias: Die zehn Gebote, wenn Word spinnt)
▼ Links zu diesem Thema

Frage

▲ Zurück zum Anfang

> Ich möchte Word neu installieren und dabei die von mir in mühsamer Arbeit getätigten Einstellungen nicht verlieren. Wo speichert Word dieses Optionen? Was muss ich sichern?

Voranbemerkungen

▲ Zurück zum Anfang

Eigentlich müssen Sie vor einer Neuinstallation von MS Word gar nichts sichern, da bei einer Neuinstallation von Word die Einstellungen, welche der Benutzer vorgenommen hat, nicht überschrieben werden.

Das ist eigentlich ganz nützlich, hat aber schon den einen oder anderen an den Rande des Wahnsinns getrieben, hoffte er doch, mit einer Neuinstallation gewisse Fehler, hinter denen er irgend eine falsche Einstellung vermutete, zu beseitigen. Und siehe da, es hat sich leider gar nichts verändert.

Wo speichert MS Word seine Einstellungen? - Was für eine harmlose und einfach erscheinende Frage!

In der Tat ist es eine sehr komplexe Frage, welche nur differenziert beantwortet werden kann. MS Word ist um Dimensionen komplexer, als ein einfaches Malprogramm, wie es z.B. *MS Paint* darstellt, wo die Einstellungen tatsächlich mit einem oder zwei Schlüsseln in der Registrierdatenbank abgedeckt werden können. Kommt hinzu, dass verschiedene Benutzer unter Einstellungen (oder Optionen) etwas anderes verstehen.

Weiter ist zu bedenken, dass viele Einstellungen bewusst indirekt oder sogar indirekt-indirekt festgelegt sind. Die Schriftgrösse z.B., wird über die **Formatvorlage** mit dem reservierten Namen «*Standard*» festgelegt. Formatvorlagen wiederum sind Bausteine, welche in einer **Dokumentenvorlage** (allenfalls sogar im Dokument selbst) gespeichert sind. Eine besondere Bedeutung bezüglich der Dokumentenvorlagen kommt der Datei mit dem Namen **Normal.dot** zu. Diese Dokumentenvorlage gibt nämlich immer dann den Takt an, wenn Sie keine anderen Vorkehrungen unternommen haben, also beim Erstellen eines neuen Dokumentes, keine spezielle Vorlage ausgewählt haben.

Für sehr eilige Besucher

- Nehmen Sie dankend zur Kenntnis, dass bei einer Neuinstallation von MS Word, solange Sie nichts löschen oder umbenennen und auch die selben Verzeichnisse verwenden, keine Einstellungen verloren gehen, weder theoretisch noch in der Praxis.

Für eilige Besucher

- Lesen Sie bitte das Kapitel Fazit am Schluss diese Skriptes..

Für Besucher, welche es genauer wissen möchten

- Ich hoffe, dass Sie nach Sichtung dieses Skriptes, sowohl was Komponenten und auch Zusammenhänge betrifft, etwas schlauer sind. Die Sache ist in der Tat so komplex, dass kein Schwein diese Informationen immer und im richtigen Zeitpunkt präsent hat.

Formatvorlagen als Trägerinnen von Einstellungen

▲ Zurück zum Anfang

Wie schon in der Einleitung angedeutet, können Formatvorlagen als Trägerinnen von Einstellungen fungieren. Entnehmen Sie der folgenden Tabelle, die Namen von einigen reservierten Formatvorlagen und deren Bedeutung. Unter der Vielzahl dieser Formatvorlagen habe ich bewusst nur jene aufgelistet, welche erfahrungsgemäss oft vergessen gehen. Der Benutzer ist sich oft nicht bewusst, dass die Formatierung von Fussnotenzahlen, Zeilennummern und vieles mehr dort festgelegt ist.

Name der Formatvorlage	Bedeutung
Absatz-Standartschriftart	Legt die Zeichenformatierung von Text fest, welcher nicht mit einer spezifischen Formatvorlage formatiert ist. Stellt die Mutter aller Zeichenformatvorlagen fest, welche darauf verweisen (und dafür sollten Sie eigentlich immer bemüht sein)
Standard	Legt die Absatzformatierung von Text fest, welcher nicht mit einer spezifischen Formatvorlage formatiert ist. Stellt die Mutter aller Absatzformatvorlagen fest, welche darauf verweisen (und dafür sollten Sie eigentlich immer bemüht sein)
Beschriftung	Legt die Standardformatierung einer Bildunterschrift etc. fest.
Dokumentstruktur	Legt fest, wie die Überschriften angezeigt werden, wenn unter Ansicht die Dokumentstruktur eingeschaltet ist.
Fußnotentext	Legt die Standardformatierung des Fussnotentextes fest.
Fußnotenzahl	Legt die Standardformatierung der Fussnotenzahl fest.
Hyperlink	Legt das Erscheinungsbild eines Hyperlinks fest.
BesuchterHyperlink	Legt das Erscheinungsbild eines besuchten Hyperlinks fest.
Umschlagadresse	Legt die Formatierung für eine Umschlagsadresse fest (Extras - Umschläge und Etiketten).
Umschlagabsenderadresse	Legt die Formatierung für eine Absenderadresse fest (Extras - Umschläge und Etiketten).
Zeilennummer	Legt die Formatierung der Zeilennummern fest (Datei - Seite einreichten - Seitenlayout - Zeilennummern).

Weitere Einstellungen, welche indirekt über Formatvorlagen getätigt werden

Hier sticht vor allem die Spracheinstellung hervor, welche einem Text zugewiesen wird.

Die Bedeutung der Standardvorlage mit dem Namen «*Normal.dot*»

▲ Zurück zum Anfang

Die Überlegungen des vorhergegangenen Kapitels sollten uns hellhörig machen. Offenbar existiert für vieles gar keine globale Einstellung. Um sicher zu gehen, bewahren Sie aber eine Kopie der Normal.dot auf, welcher trotz allem eine zentrale Bedeutung zu kommt.

Diese ominöse Datei mit dem Namen *Normal.dot*, enthält auch die meisten Bausteine, welche man in MS Word anlegen kann. Der Speicherort von Word Bausteinen ist nicht Gegenstand dieses Skriptes. Wenn Sie näheres über Bausteine erfahren möchten, schauen Sie sich den Link an, welcher am Schluss dieses Skriptes eingeblendet ist.

In diesem Sinne hier nur eine ganz kurze Zusammenfassung:

Bausteine, welche in der "Normal.dot" gespeichert sind

- **Formatvorlagen:** Beachten Sie, dass auch Dokumente Formatvorlagen enthalten können. Die Definitionen von gleichlautenden Formatvorlagen in der spezifischen Dokumentenvorlage, in der *Normal.dot* und im Dokument können sich unterscheiden.
- **AutoTexte** (Beachten Sie, dass auch andere Dokumentenvorlagen AutoTexte enthalten können.)
- Gewisse **AutoKorrektur-Einträge** allerdings nur bestimmte. Lesen Sie dazu das Kapitel AutoKorrektur-Einträge, im Skript *Word-Bausteine* auf welches am Ende dieser Seite verwiesen wird.
- **Makros:** (unter anderem auch allfällige Automationsroutinen (*AutoNew*, *AutoOpen*, etc.) Beachten Sie, dass auch Dokumente Makros enthalten können.
- Vom Benutzer veränderte integrierte **Symbolleisten** und Symbolleisten, welche benutzerdefiniert sind. Beachten Sie, dass auch Dokumente Symbolleisten enthalten können.
- Benutzerdefinierte **Tastenkürzel** (*Shortcuts*)
- Einstellungen, welche die **Silbentrennung** betreffen
- Der **Zoom-Faktor**, indem ein Dokument angezeigt wird

Bausteine, welche nicht in der Normal.dot oder einer anderen Dokumentenvorlage gespeichert sind

- Die meisten AutoKorrektor-Einträge.
- Vom Benutzer zugefügte Wörter zum Wörterbuch für die Rechtschreibprüfung.
- Benutzerdefinierte Etiketten.
- Word-weite, globale Einstellungen. (siehe nächstes Kapitel)

Anwendungsweite Einstellungen in der Registrierdatenbank

▲ Zurück zum Anfang

Bleiben noch die Einstellungen, welche nicht an eine Dokumentenvorlage oder einer der Bausteine einer Dokumentenvorlage gebunden sind. Dabei handelt es sich um jene Einstellungen, welche Word-weit gültig sind, also **Word-globale** Einstellungen. Diese Optionen sind in der Registrierdatenbank gespeichert.

Das Hauptaugenmerk richtet sich dabei auf einen einzigen Schlüssel in der Registrierdatenbank, welcher in einer sehr langen (binären) Zeichenfolge eine Vielzahl dieser Optionen enthält. Es handelt sich dabei um folgenden Schlüssel:

`HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data`
Word 97 bzw.
`HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Data`
Word2000

Diese Informationen erscheinen dem Betrachter unstrukturiert und sie liegen im binären Format vor. Damit können Sie kaum etwas aus diesen Informationen herauslesen und sie können diese Einstellungen schon gar nicht editieren.

Dieser Schlüssel enthält zwei Werte, welche in verschachtelter Form die Word-globalen Optionen beinhalten. Die Namen dieser zwei Werte lauten **«Settings»** und **«Toolbars»**.

Im folgenden finden Sie eine Auflistung der Optionen, welche in diesem Schlüssel gespeichert sind, soweit ich dies identifizieren konnte.

Der Wert «Settings»

- Viele Optionen, welche Sie im Word-Dialog **Extras - Optionen...** wiederfinden.
- Die Liste der **zuletzt geöffneten Dateien**, welche Sie im Dialog *Datei* vorfinden.
- Welche **Ansicht** bei Verlassen von Word aktiv war (Normalansicht, Seitenlayoutansicht, etc)
- Die **Fenstereinstellung** beim Verlassen von Word (*Maximiert*, *Normal* oder *Minimiert*) und falls *Normal* zutreffen sollt, die Position des Fensters.
- Verweise, welche Sie unter **Extras - Vorlagen und Add-Ins** oder im VBA-Editor unter **Extras - Verweise...** setzen.
- Die **zuletzt benutzten Adressen**, wenn Sie ex Word mit einem MAPI-fähigen Adressbuch arbeiten.

Der Wert «Toolbars»

Dieser Schlüssel enthält Informationen zur Sichtbarkeit und Position von Symbolleisten, beim Beenden von Word. Die Symbolleisten selbst sind allerdings in Dokumentenvorlagen (z.B. Normal.dot) oder in Dokumenten gespeichert!

Der Schlüssel «Options»

Anders als der Schlüssel mit dem Namen «*Data*», können die Werte des Schlüssels «*Options*» sehr wohl eingesehen und editiert werden. Dieser Schlüssel kann viele Einträge haben oder aber nur wenige. Falls ein bestimmter Eintrag nicht vorhanden ist, kommt der Standardwert zum tragen. Will man einen anderen Wert als der Standardwert aktivieren, muss man zuerst den entsprechenden Eintrag anlegen und ihm anschliessend den gewünschten Wert zuweisen. Will man den Standard wieder herstellen, kann man den Standardwert eintragen oder den Eintrag einfach wieder löschen (wobei nun nicht der Anschein aufkommen soll, man soll den ganzen Schlüssel löschen, sondern lediglich einen bestimmten Eintrag). Viele Einträge in diesem Schlüssel sind aber nicht dazu gedacht, dass man eine Änderung direkt in der Registrierdatenbank vornimmt, wenn dies auch möglich ist. In erster Linie ist dafür der Dialog **Extras - Optionen...** in Word zuständig; und in zweiter Linie ein Hilfsmittel von Microsoft mit dem Namen **Support8.dot Word 97** bzw. **Support9.dot Word2000**. Wie Sie diese Dateien auftreiben können, wird später besprochen.

Hier eine Auswahl der Optionen, welche letztlich in diesem Schlüssel gespeichert sind. Beachten Sie beim Studium der Tabelle bitte folgendes:

- Die Liste der Einträge ist **nicht** komplett.
- Das Gros dieser Einträge muss nicht zwingend in Ihrer Registrierdatenbank vorkommen. In der Regel finden Sie nur einen entsprechenden Eintrag, wenn dieser vom Standard abweicht. Diese Standards sind von Microsoft vorgegeben und gespeichert in einer Pragrammbibliothek (*DLL*).
- Die meisten Einträge müssen nicht direkt in der Registrierdatenbank editiert werden. Einige davon können über den Dialog ***Extras - Optionen...*** in Word eingestellt werden, andere wiederum dank einem Makro, welche Microsoft bereit stellt. Die Bezugsquelle dieses Makro wird gleich anschliessend angegeben.
- Seien Sie vorsichtig im Umgang mit diesen Einstellungen: ungültige oder stark überrissene Werte, können die Funktionstüchtigkeit von Word - ja im Extremfall den Betrieb des ganzen Systems - negativ beeinflussen. Sollten Sie diesbezüglich einen Verdacht haben, schliessen Sie Word und benennen Sie den ganzen Schlüssel von ***Options*** in *Options1* um. Word wird dann den Schlüssel beim nächsten Start neu initialisieren.

Eintrag	Bedeutung
AutoSpell **AutoGrammer**	Einstellungen bezüglich der interaktiven Rechtschreibprüfung.
AutoSave-Path	Der Pfad für die Speicherung der Wiederherstellungsdateien.
BackgrondSave **BackgrondPrint**	Einstellungen bezüglich "Speichern im Hintergrund" und "Drucken im Hintergrund".
StandardFormat	Das Standardformat der Word-Dateien.
Picture-Path	Der Standardpfad für den Dialog "Einfügen - Grafik".
Bak-Extension	Die Dateierweiterung für Sicherungskopien.
Doc-Path	Der Standard-Ordner, in welchem Word-Dateien gespeichert werden.
Startip-Path	Der Pfad des "Startup"-Ordners (Dort befinden sich die globalen Dokumentenvorlagen, welche bei jedem Start von Word geladen werden)

Hilfsmittel von Microsoft

Mit einem Makro welches sich in einer Dokumentenvorlage mit dem Namen **Support8.dot** **Word 97** bzw. **Support9.dot** **Word2000** befindet, können Sie gewisse Einstellungen in der Registrierdatenbank gefahrlos verändern.

Sie finden diese Datei auf Ihrer Office-CD, Der Ablauf ist nach einem Doppelklick auf die Dokumentenvorlage Dialog-geführt.

Word-Bausteine

▲ Zurück zum Anfang

Bausteine sind ein weiteres wichtiges Stichwort zu diesem Thema. Dieses Gebiet ist so umfangreich, dass es in einem anderen Skript abgehandelt wird. Dieses Skript finden Sie unter dem folgenden Link:

► Wo werden Bausteine wie Formatvorlagen, AutoTexte, Symbolleisten, AutoKorrektur- und Benutzerwörterbucheinträge gespeichert?

Benutzerdefinierte Etiketten

▲ Zurück zum Anfang

MS Word bietet dem Benutzer auch die Möglichkeit Etikette mit speziellen Massen zu definieren. Dies geschieht in Word über den Dialog *Extras - Umschläge und Etikette... - Etiketten*. Die Definitionen dazu werden im folgenden Schlüssel in der Registrierdatenbank abgelegt:

`HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Custom Labels`
Word 97 bzw.

`HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Custom Labels`
Word2000

Um diese Informationen zu portieren, gehen Sie bitte wie folgt vor:

1. Starten Sie den Registrierungseditor.
2. Arbeiten Sie sich zum genannten Schlüssel vor und markieren Sie diesen.
3. Im Hauptmenü wählen Sie nun Registrierung - Registrationsdatei exportieren....
4. Vergeben Sie nun einen aussagekräftigen Dateinamen z.B. **Etikett.dat**. Die Dateierweiterung "dat" ist dabei zwingend.
5. Kopieren Sie die entstandene Datei auf eine Diskette.
6. Vergewissern Sie sich, dass nur diesen Schlüssel und genau diesen Schlüssel gesichert haben. Sie können die entstandene *.reg-Datei mit jedem Editor, z.B. Notepad einsehen.
7. Beim Zielrechner führen Sie einen Doppelklick auf diese Datei aus und bestätigen die obligate Rückfrage von Windows.
8. Damit sollten diese Definitionen auf dem Zielrechner zur Verfügung stehen.

Fazit

▲ Zurück zum Anfang

- Viele Einstellungen sind in MS Word indirekt oder sogar indirekt-indirekt festgelegt. Dies macht die Sache für den Einsteiger nicht einfacher, erhöht die Flexibilität und die Produktivität der Anwendung aber gewaltig. Für den erfahrenen Benutzer erhöht sich die Transparenz sogar durch dieses Konzept.
- Versierte Textverarbeitungsprogramme gehören zu der komplexesten Software, welche je erschaffen wurde. Dies spiegelt sich u.a. auch in der Vielfalt von Einstellungen wider.
- Bei einer Neuinstallation von MS Word werden keine Einstellungen überschrieben, es sei denn, Sie löschen oder verschieben Dateien oder installieren in ein anders Verzeichnis, als das ursprüngliche.
- Möchten Sie trotzdem Ihre Umgebung sichern, dann suchen, kopieren und verwaren Sie folgende Dateien:
  - Die **Normal.dot**.
  - **Benutzerdefinierte Dokumentenvorlage**, welche Ihnen wichtig sind. Am besten Sie kopieren gleich das ganze Vorlagenverzeichnis. Neben dem Standardvorlagenverzeichnis ist allenfalls noch ein Arbeitsgruppenvorlagenverzeichnis vorhanden.
  - Sämtliche Dateien mit der Dateierweiterung ***.acl** (Das ist die hauptsächliche Datenbasis für die AutoKorrektur.)
  - Sämtliche Dateien mit der Dateierweiterung ***.dic** (Das sind die Benutzerwörterbücher.)
  - Nochmals ans Herz legen möchte ich Ihnen, die Bedeutung von Formatvorlagen, welche in der Regel in Dokumentenvorlagen und in Einzelfällen auch in Dokumenten gespeichert sind. Wenn Sie sich also grosse Mühe gegeben haben, diese geschickt und nach Ihren Wünschen anzupassen, sichern Sie diese Arbeit, indem Sie die Dokumentenvorlage, welche Trägrein dieser Formatvorlagen ist, gut aufbewahren. Formatvorlagen sind schon fast das «*A & O*» eines Textverarbeitungssystems. Ich persönlich habe noch nie verstanden, wieso Microsoft keinen effizienteren Weg schafft, um Formatvorlagen zu definieren bzw. zu editieren. Zwar lässt die Parametrisierung von Formatvorlagen kaum Wünsche übrige; die Tatsache dass man aber immer nur eine Formatvorlage vor den Augen hat, und diese über eine ganze Kette von Dialogen anpassen muss, kann es ja wohl nicht sein. Das Konkurrenzprodukt *WordPerfect* bietet da wesentlich bessere Alternativen und ein *Cascading Style Sheet* (CSS) als Bestandteil von HTML lässt sich ganz bestimmt auch effizienter unterhalten, als dieses gut gemeinte, aber überholte Dialogspiel in MS Word.

Installationswerte wieder herstellen (Alias: Die zehn Gebote, wenn Word spinnt)

▲ Zurück zum Anfang

Gelangen Sie zum Eindruck, dass Ihre Word-Installation nicht mehr richtig arbeitet, weil gewisse Installationswerte defekt oder falsch sind, dann können Sie dem mit folgenden Aktionen entgegenwirken.

Die meisten genannten Aktionen können auch unabhängig voneinander geschehen, ich habe mich aber bemüht eine gewisse Chronologie in die Sache einzubringen. Im einzelnen und individuellen Fall, mag es aber Sinn machen, von dieser Reihenfolge abzuweichen.

Führen Sie bitte die folgenden Schritte aus:

1. Schliessen Sie Word
2. Lassen Sie nach der Standardvorlage mit dem Namen **Normal.dot** suchen und nennen Sie diese Datei in *Alt.dot* um. Sollten Sie mehr als eine *Normal.dot* finden (in verschiedenen Verzeichnissen) nennen Sie im Zweifelsfall alle um.
3. Im Registrierungseditor nennen Sie folgenden Schlüssel nach **Data1** um:
   HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data
   Word 97 bzw.

   HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Data
   Word2000
4. Immer noch im Registrierungseditor nennen Sie auch folgenden Schlüssel nach **Options1** um:

   HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Options
   Word 97 bzw.

   HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Options
   Word2000
5. Nenne Sie sämtlich Dateien mit der Dateierweiterung ***.acl** um.
6. Nenne Sie sämtlich Dateien mit der Dateierweiterung ***.dic** um.
7. Entfernen Sie alle Dateien im Verzeichnis mit dem Namen **StartUp** (dabei handelt es sich um ein Unterverzeichnis des Office-Verzeichnisses). Schieben Sie diese Dateien in ein neutrales Verzeichnis, welches Sie gegebenenfalls neu erstellen. Damit verhindern Sie, dass diese Dateien beim Start von MS Word automatisch mitgeladen werden.
8. Löschen Sie alle temporären Dateien auf ihrem PC.
9. Starten Sie MS Word über ***Start - Ausführen*** mit einem speziellen Schalter nämlich wie folgt:
   "x:\ihr office pfad\winword.exe" /r

   Selbstverständlich müssen Sie den Laufwerkbuchstaben und den Pfad noch anpassen. Beachten Sie auch, dass lange Pfadnamen in Gänsefüsschen stehen müssen und dass vor dem Schalter **"/r"** zwingend ein Leerschlag sein muss.
10. Schliessen Sie nun Word und starten Sie Word auf die übliche Art und Weise.

Mit diesen Aktionen, welche übrigens alle umkehrbar sind, sollt MS Word wieder beinahe so daher kommen, als hätten Sie die Anwendung neu installiert.

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Wo werden Bausteine wie Formatvorlagen, AutoTexte, Symbolleisten, AutoKorrektur- und Benutzerwörterbucheinträge gespeichert?

Microsoft

Englisch

► WD97: Where Settings Are Stored in the Registry
► WD2000: Where Settings Are Stored in the Registry
► WD97: How to Reset User Options and Registry Settings
► WD2000: How to Reset User Options and Registry Settings

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Optionen.htm

René Probst, Oktober 2000

Änderungen

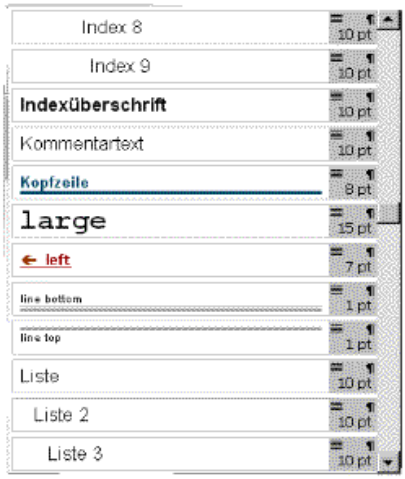
- --- nil ---

Wie kann ich meine Formatvorlagen übersichtlich organisieren?

◄ Zurück zum Hauptindex

Problem

> **Das Kombinationsfeld Formatvorlagen auf der Symbolleiste Format finde ich für meine Zwecke nicht praktisch. Die Darstellungsweise fordert mir viel *Scroll*-Arbeit ab. Zudem sammeln sich mit der Zeit Formatvorlagen an, welche ich gar nicht oder nur sehr selten benötige. Kommt hinzu, dass diese Auflistung der Formatvorlagen nicht alphabetisch ist.**

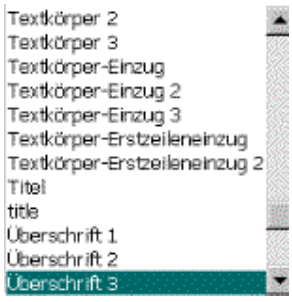


Das integrierte Kombinationsfeld zur Auswahl der Formatvorlagen ist zwar gefällig und informativ, fordert aber viel "Scroll-Arbeit" ab.

Antwort

Versuchen wir die Eindrücke etwas zu ordnen:

- Es handelt sich beim fraglichen Steuerelement um ein Kombinationsfeld, welches graphisch dargestellte Einträge zulässt. Sie können die Darstellung dieses Steuerelementes wohl nicht ändern, es sei denn, Sie setzen Word2000 ein (siehe weiter unten).
- Das viel kritisierte Steuerelement hat immer hin den Vorteil, dass es dynamisch die angezeigte Formatvorlage anpasst, wenn Sie die Einfügemarke an einen Ort stellen an welchem eine andere Formatvorlage aktiv ist. Diese Potenz erreichen Sie auch mit VBA in Word 97 nicht. Allenfalls könnten Sie dies in Word2000, welches auf Ereignisse im Dokument reagieren kann, mit entsprechenden Programmierkenntnissen erreichen.
- Tatsächlich sind die Namen der Formatvorlagen in **Word 97** nicht alphabetisch sortiert, in **Word2000** allerdings schon.
- Ebenfalls nur in Word2000 gibt es eine Option, welche Ihnen doch einiges an Scroll-Arbeit erspart. Sie finden diese Option im Word-Dialog ***Extras - Anpassen... - Optionen***. Wenn Sie dort das Häkchen bei **Schriftnamen in Schriftart** wegmachen, wird auch die Liste der Formatvorlagen bedeutend kompakter angezeigt.



Schon wesentlich kompakter präsentiert sich der selbe Dialog, wenn die grafische Anzeige ausgeschaltet wird. Das ist aber ausschliesslich in Word2000 möglich.

- Sie können zwar die Liste der Formatvorlagen ausmisten, in dem Sie eine neue Vorlage erstellen und über den Dialog ***Extras - Vorlagen und Add-Ins - Organisieren*** nur diejenigen Formatvorlagen übernehmen, welche Sie wirklich gebrauchen. Dieser Erfolg ist allerdings nicht sehr dauerhaft. Sie werden es nicht verhindern können, dass sich in Zukunft wieder Ballast ansammelt. Dieser Ballast ist meistens nicht selbst fabriziert, er stammt aus "fremden" Dokumenten oder Vorlagen, welche sie irgendwann geöffnet haben.
- Ein weiterer wichtiger Grund, wieso sich mit der Zeit viele Formatvorlagen ansammeln ist folgender: wenn Sie z.B.ein Inhaltsverzeichnis erstellen, dann kommen zu den bereits vorhandenen Formatvorlagen - implizit - noch die Vorlagen mit den Namen Verzeichnis 1-n dazu. Das gleich trifft für *Einfügen - Beschriftung* etc. zu. Diese Formatvorlagen machen zwar Sinn, allerdings werden Sie diese kaum jemals aktiv zuweisen und schon gar nicht in einem Dokument ohne Inhaltverzeichnis oder Beschriftungen. Dies wäre ja auch nicht ratsam. Es kommt allerdings manchmal den Wunsch auf, auch diese Formatvorlagen neu zu formatieren. Zu diesem Zweck müssten diese allerdings nicht in dieser feudalen Art angezeigt werden, wie es unter Word der Fall ist. Diese grafische Anzeige macht ja nur dann Sinn, wenn Sie das Aussehen einer Formatvorlage, welche nicht schon eh im Dokument vor Ihrer Nase steht, erkennen möchten.

Lösungsvarianten

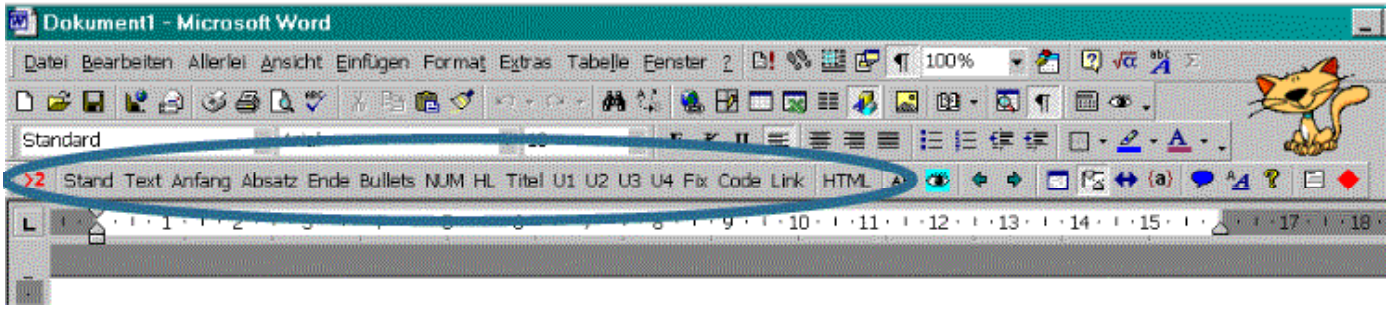
Ich schlage Ihnen vor, sich selbst ein Menü, oder je nach Geschmack, eine Symbolleiste mit Ihren Lieblingsformatvorlagen, à la carte, zusammen zu stellen. Sie erreichen dies auf einfache Weise, ohne eine Zeile VBA-Code dafür schreiben zu müssen. Der Vorteil eines Menüs liegt darin, dass es Platz für fast eine beliebige Anzahl von Formatvorlagen bietet. Bei einer Symbolleiste ist dieser Punkt kritischer. Dafür sind die Schaltflächen in einer Symbolleiste noch etwas schneller zugänglich als Menüpunkte. Um Ihr eigenes Formatvorlagenmenü zu erstellen, gehen Sie wie folgt vor:

Eigenes Menü mit Formatvorlagen erstellen

Ganz ohne zu Programmieren, können Sie sich in wenigen Minuten ein Menü mit genau jenen Formatvorlagen zusammen stellen, welche Sie oft und aktiv benutzen.

1. Wählen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen...***.
2. Wechseln Sie in den Reiter ***Befehle***.
3. Markieren Sie in der linken Liste **Kategorien:** den Eintrag **Neues Menü**.
4. Wählen Sie nun auch in der rechten Liste **Befehle:** den Eintrag **Neues Menü**.
5. Ziehen Sie diesen Eintrag auf die Menüleiste oder einer Symbolleiste Ihrer Wahl.
6. Markieren Sie nun wieder auf der linken Liste Kategorien: den Eintrag **Formatvorlagen**.
7. Wiederum auf der rechten Liste erscheinen nun alle Verfügbaren Formatvorlagen.
8. Ziehen Sie nun nacheinander Ihre Lieblingsvorlagen in das von Ihnen neu erstellte Menü.
9. Im Anschluss können Sie den Namen des von Ihnen erstellte Menüs abändern, in dem Sie immer noch im Anpassungsmodus einen Rechtsklick auf das Menü ausführen und unter **Namen** eine Bezeichnung Ihrer Wahl eingeben.
10. Auf die gleiche Weise können Sie anschliessend die Namen der einzelnen Menüpunkte etwas abkürzen. Es ist sicher unnötig, wenn dort hinter jeder Formatvorlage noch das Wort Formatvorlage steht.
11. Wenn Sie mit Ihrem Werk zufrieden sind, schliessen Sie den Dialog mit der Schaltfläche ***Schliessen*** ab.

Eigene Symbolleiste mit Formatvorlagen erstellen



Am bequemsten ist der Zugriff auf Formatvorlagen, wenn Sie diese auf einer Symbolleiste aufreihen. Wenn Sie die sichtbare Bezeichnung der Formatvorlagen abkürzen, passen viele Formatvorlagen auf eine Symbolleiste.

Um Ihr eigene Symbolleiste mit Ihren Lieblingsformatvorlagen zu erstellen, gehen Sie wie folgt vor:

1. Wählen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen***.
2. Wechseln Sie in den Reiter ***Symbolleisten***.
3. Betätigen Sie die Schaltfläche ***Neu...***.
4. Vergeben Sie einen Namen Ihrer Wahl für die neue Symbolleiste und betätigen Sie die Schaltfläche ***Ok***.
5. Wechseln Sie nun in den Reiter ***Befehle***.
6. Markieren Sie nun in der linken Liste **Kategorien:** den Eintrag **Formatvorlagen**.
7. In der rechten Liste erscheinen nun alle Verfügbaren Formatvorlagen.
8. Ziehen Sie nun nacheinander Ihre Lieblingsvorlagen in die von Ihnen erstelle Symbolleiste.
9. Immer noch im Anpassungsmodus, können Sie nun die Namen der einzelnen Schaltflächen etwas abkürzen. Es ist sicher unnötig, wenn dort hinter jeder Formatvorlage noch das Wort Formatvorlage steht. Sie erreichen dies, indem Sie einen Rechtsklick auf die Schaltfläche ausüben und das Feld **Name:** abändern.
10. Wenn Sie eher eine Person sind, welche das Umfeld visuell aufnimmt, können Sie im selben Kontextmenü das Häkchen beim Menüpunkt **Standard** statt *Nur Text (immer)* setzen und über den Menüpunkt **Schaltflächensymbol ändern** der Schaltfläche eine Ikone Ihrer Wahl zuweisen.
11. Wenn Sie mit Ihrem Werk zufrieden sind, schliessen Sie den Dialog mit der Schaltfläche ***Schliessen*** ab.

Anmerkung

- Falls Word Sie beim Verlassen anfragt, ob die Standardvorlage **Normal.dot** gespeichert werden soll, antworten Sie mit ***Ja*** damit das neue Menü oder die neue Symbolleiste auch gespeichert wird.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/FVOrg.htm

René Probst, Oktober 1999

Änderungen
- April 2000 - Die Sache mit eingefügten Grafiken verdeutlicht.

Wie referenziere ich das jeweilige Kapitel in der Kopf- oder Fusszeile?

◄ Zurück zum Hauptindex

Aufgabenstellung

> **Sie möchten Ihr Dokument so einrichten, dass in der Kopf- oder Fusszeile die Überschrift des Kapitels erscheint, welches auf der jeweiligen Seite abgehandelt wird.**

Lösung

Sie erreichen dies mit der Feldfunktion **FVRef** (FVRef steht für Formatvorlagenreferenz). Dieser Begriff gilt für **Word 97**. In **Word2000** heisst die selbe Feldfunktion **StyleRef**. Es handelt sich dabei aber um exakt das selbe. Word2000 verwendet für Feldnamen eben nur noch die englischen Bezeichnungen.

Wie Sie vielleicht erraten haben, läuft die ganze Sache über Formatvorlagen ab. Möchten Sie also z.B. auf die Überschriften auf höchster Ebene verweisen, benutzen Sie diese Feldfunktion im Zusammenhang mit der Formatvorlage *Überschrift 1*.

Die Feldfunktion kennt verschiedene **Schalter** (Switches) mit denen Sie Verhalten und Aussehen der Referenz beeinflussen können. Wenn Sie den Office-Assistenten bemühen und nach "FVRef" suchen lassen, gelangen Sie zu folgendem Hilfetext:

> ***{ FVREF Erkennungszeichen [Schalter] }***
> *Fügt Text ein, der mit der angegebenen Formatvorlage formatiert ist. Wenn das **FVRef**-Feld in eine Kopf- oder Fußzeile eingefügt ist, druckt es die erste oder letzte Textstelle, die auf der aktuellen Seite mit der angegebenen Formatvorlage formatiert ist, so dass Sie Kopf- oder Fußzeilen im Wörterbuchformat drucken können.*

Der Hilfetext fährt dann fort mit der Beschreibung aller möglichen Schalter und vermittelt auch noch zusätzliche Informationen, welche dieses Thema wohl erschöpfend abhandeln. Sie können übrigens auch mehrere dieser Felder kombinieren, um an Ihr Ziel zu gelangen.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/FVRef.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Word findet bei der Generierung des Abbildungsverzeichnisses keine Einträge

◄ Zurück zum Hauptindex

Symptome

> **Den Versuch ein Abbildungsverzeichnis zu generieren quittiert Word mit der Fehlermeldung:**
>
> ***Fehler! Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.***

Ursache

Werden Grafiken mit Beschriftungen über den Text gelegt, hat dies zur Folge, dass die Beschriftung von der Word-Anwendung automatisch in ein Textfeld eingebunden wird. Nach dem Objektmodell von Word handelt es sich bei Textfeldern aber um spezielle Teile des Dokumentes. Genau diese Teile werden nicht nach Beschriftungen durchsucht, wenn ein Abbildungsverzeichnis erstellt wird.

Abhilfe

Sie können nachträglich die Textfelder in Positionsrahmen konvertieren. Gehen Sie dabei wie folgt vor:

1. Markieren Sie das Textfeld am Rand.
2. Wenn Sie den Rahmen des Textfeldes markiert haben, führen Sie einen ***Rechtsklick*** auf diesen aus.
3. Im erscheinenden Kontextmenü wählen Sie jetzt den Eintrag ***Textfeld formatieren***
4. Wählen Sie nun den Reiter ***Textfeld***.
5. Nun betätigen Sie die Schaltfläche ***Konvertieren zu Positionsrahmen...***.
6. Word wird Sie jetzt warnen, dass gewisse Formatierungen verloren gehen können, Dieser Hinweis ist aber nicht auf den Text gemünzt. Ignorieren Sie also diese Warnung.
7. Haben Sie das beschrieben Verfahren auf alle Beschriftungen angewendet, können Sie frischen Mutes das Abbildungsverzeichnis erstellen.

Sollte dies zu aufwendig sein, da das Dokument über sehr viele Beschriftungen verfügt, lohnt sich wohl der Einsatz eines Makros. Testen Sie vielleicht zuerst mit einer Kopie des Dokumentes, ob der VBA-Code auch bei Ihnen die diese Aufgabe zuverlässig und ohne Nebenwirkungen erledigt.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub TextBoxesToFrames()
  svShowFieldCode = ActiveWindow.View.ShowFieldCodes
  ActiveWindow.View.ShowFieldCodes = True
  Application.ScreenUpdating = False
  Dim TB As Shape
  For Each TB In ActiveDocument.Shapes
    If TB.Type = msoTextBox Then
      If InStr(TB.TextFrame.TextRange, Chr(19) + " SEQ") > 0 Then
        TB.ConvertToFrame
      End If
    End If
  Next
  Selection.HomeKey Unit:=wdStory
  ActiveWindow.View.ShowFieldCodes = svShowFieldCode
  Application.ScreenUpdating = True
End Sub
```

(Befinden sich Beschriftungen in Textfeldern, werden diese Textfelder in Positionsrahmen konvertiert.)

Vorbeugung

Sie können diese Problematik umgehen indem Sie einen der folgenden Punkte beachten:

- Deaktivieren Sie beim Einfügen der Grafik, also bevor sie die Beschriftung einfügen, die Option **Über den Text legen**.
- Achten Sie darauf, dass die Grafik **nicht** markiert ist, wenn Sie die Beschriftung einfügen.

Anmerkungen

- Diese Problematik hat sich in Word2000 entschärft, weil dort die Option *Über den Text legen*, standardmässig ausgeschaltet ist.
- Da ich das Objektmodell von Word gut kenne, verstehe ich ein wenig, in welches Dilemma sich Microsoft bezüglich dieser Angelegenheit manövriert hat. Trotzdem gehören für mich Beschriftungen und Abbildungsverzeichnis eng zusammen. Deshalb wundert es mich doch sehr, dass Microsoft diese Sache nicht besser abgestimmt hat. Man überlege sich mal, wie viel wertvolle Arbeitszeit mit einem solchen Patzer verloren geht.

Links zu diesem Thema

Microsoft

Deutsch

► Im Abbildungsverzeichnis fehlt Beschriftung aus Textfeldern

Englisch

► WD97: Table of Figures (TOF) Missing Captions from Text Boxes

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Beschr.htm
René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich ein Inhaltsverzeichnis in ein anderes Word-Dokument übernehmen

◄ Zurück zum Hauptindex

Problem

> **Wenn ich mir mit Word ein Inhaltsverzeichnis generieren lasse und anschliessend dieses Inhaltsverzeichnis kopiere, um es in ein anderes Dokument einzufügen, erscheint dort das Inhaltsverzeichnis zwar. Sobald ich aber die Felder aktualisiere oder das Dokument schliesse und wieder öffne, erscheint an dieser Stelle nur noch folgende Fehlermeldung:**
>
> ***Fehler! Es wurden keine Einträge für das Inhaltsverzeichnis gefunden.***

Ursache

Diese Beobachtung ist richtig. Beim Inhaltsverzeichnis handelt es sich nicht um statischen Text sondern um eine dynamische Feldfunktion. Immer wenn die Felder aktualisiert werden, wird auch das Inhaltsverzeichnis basierend auf den aktuellen Überschriften neu aufgebaut. Im Dokument, in welches Sie diese Feldfunktion hinein kopieren, gibt es aber keine Überschriften bzw. andere Überschriften. Somit kann diese Mimik so gar nicht funktionieren.

Lösung

Das Problem lässt sich auf einfache Art und Weise umgehen. Gehen Sie wie folgt vor:

1. Kopieren Sie das Inhaltsverzeichnis wie gewohnt in die Zwischenablage.
2. Fügen Sie den Inhalt der Zwischenablage in das neue Dokument ein.
3. Markieren Sie das Verzeichnis am neuen Ort.
4. Betätigen Sie unmittelbar danach die Tastenkombination **[Strg + Umschalt + F9]**.
5. Sollte das Inhaltsverzeichnis nicht die gewünschte neutrale Formatierung aufweisen, drücken Sie anschliessend noch die Tastenkombination **[Strg + Leertaste]** um das Zeichenformat zurück zu setzen und die Tastenkombination **[Strg + Q]** um das Absatzformat zurück zu setzen.

Damit haben Sie die dynamische Feldfunktion durch statischen Text ersetzt. Das Inhaltsverzeichnis stellt nun eine Momentaufnahme dar und wird sich nicht mehr verändern oder mit einer Fehlermeldung aufwarten.

VBA-Code

Folgendes Makro kann dazu verwendet werden, ein Inhaltsverzeichnis in ein anderes Dokument zu exportieren. Der Prozess fragt Sie zuerst, wie viele Überschrittsebenen berücksichtigt werden sollen. Der Code erstellt dann im Quelldokument - unsichtbar für Sie - ein temporäres Inhaltsverzeichnis auf Grund Ihrer Angabe. Dieses Verzeichnis wird dann in ein vom Makro neu erstelltes Dokument eingefügt und wie oben beschrieben fixiert und stellt damit eine Art *Resumée* Ihres Skriptes dar.

```
Sub Resumée()
  TMName = "SW20000526"
  Titel = "Resumée"
  Dim oDoc As Document, nDoc As Document, oRange As Range
  Flag = False
  While Flag = False
    x = InputBox("Wieviele Ebenen sollen es dann sein?", Titel)
    If x = "" Then End
    If Not IsNumeric(x) Then
      MsgBox "Geben Sie 1-9 ein.", vbExclamation, Titel
    Else
      If x < 1 Or x > 9 Then
        MsgBox "Geben Sie 1-9 ein.", vbExclamation, Titel
      Else
        Flag = True
      End If
    End If
  Wend
  Set oDoc = ActiveDocument
  oDoc.Bookmarks.Add Name:=TMName, _
     Range:=ActiveDocument.Bookmarks("\EndOfDoc").Range
  oDoc.Content.InsertAfter vbCrLf
  oDoc.Fields.Add _
     Range:=oDoc.Bookmarks("\EndOfDoc").Range, Type:=wdFieldTOC, _
           Text:="\o " & Chr(34) & "1-" & x & Chr(34) & " \n", _
              PreserveFormatting:=False
  Set oRange = _
     oDoc.Range(oDoc.Bookmarks(TMName).Range.Start, _
        End:=ActiveDocument.Bookmarks("\EndofDoc").Range.End)
  oRange.Fields.Unlink
  oRange.Copy
  Set nDoc = Documents.Add
  nDoc.Range.Paste
  oRange.Delete
  oDoc.Bookmarks(TMName).Delete
End Sub
```

(Exportiert das Inhaltsverzeichnis des aktiven Dokumentes in ein neues Dokument.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/TOCExp.htm

René Probst, Juni 2000

Änderungen

- --- nil ---

Wie kann ich mit MS Word Lückentext erstellen und handlich zwischen Aufgabenstellung und Lösung umschalten?

◄ Zurück zum Hauptindex

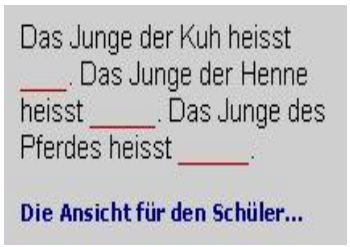
Frage

> Ich möchte einen Text erstellen, welcher Lücken aufweist, welche die Schüler später ausfüllen sollen. Wenn ich den Text, welcher verborgen bleiben soll, mit dem Zeichenformat *Verborgen* bzw. *Ausgeblendet* formatiere gelingt dies zwar, allerdings werden dann auch beim Ausdruck die Lücken eliminiert.

Word-integrierte Lösung

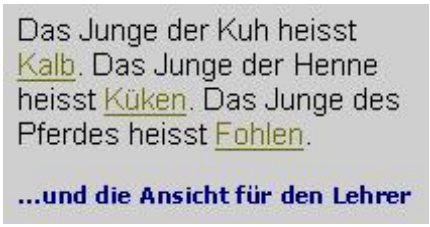
Angenommen Sie drucken den Text auf weisses Papier aus, können Sie die Textstellen, welche im Ausdruck als Lücken erscheinen sollten, mit der Farbe ***Weiss*** formatieren und dann ausdrucken. Selbstverständlich können Sie für diese Formatierung auch eine eigens erstellte Formatvorlage einsetzten.

Komfortlösung mit VBA



Es geht aber auch um einiges komfortabler und funktioneller. Dazu schlage ich die folgende VBA-Routine vor.

Die Routine richtet für Sie einen neuen **Schaltknopf** auf der Symbolleiste mit der Bezeichnung **Standard** ein. Dank diesem Ein-/Ausschalter können Sie mittels einfachem Knopfdruck, zwischen Aufgabestellung und Lösung **umschalten**. Die gewünschte Ansicht können Sie dann auch problemlos und auf üblichem Wege ausdrucken.



Die Mimik setzt zwei Formatvorlagen ein, welche ebenfalls automatisch für Sie eingerichtet werden. Wenn Sie es wünschen, können Sie diese beiden Formatvorlagen dann noch Ihren Wünschen entsprechend und zwar manuell anpassen.

So bringen Sie die Lösung zum Laufen

1. Öffnen Sie entweder die Standardvorlage mit dem Namen **Normal.dot** bzw. jene Dokumentvorlage, auf welcher derartige Schriftstücke von Ihnen basieren. Am besten Sie öffnen die Dokumentvorlage (DOT) in Word über **Datei - Öffnen**.
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
3. Dort angekommen, wählen Sie vom Hauptmenü **Einfügen - Modul**.
4. In das Fenster, welches sich geöffnet hat kopieren Sie den unten abgedruckten VBA-Code in das leere Fenster, welches sich geöffnet hat.
5. Kehren Sie nun in das Dokumentfenster zurück und lassen Sie über **Extras - Makro - Makros...** das Makro mit dem Namen **LueckentextEinrichten** ausführen.. Dieser Vorgang muss grundsätzlich genau einmal passieren.
6. Speichern Sie nun die Dokumentvorlage und schliessen Sie diese.
7. Erstellen Sie nun ein neues Dokument auf Basis dieser Vorlage. Sie sollten feststellen können, dass sich auf der Symbolleiste mit der Bezeichnung **Standard** ein neues Symbol eingenistet hat, welches das Aussehen einer **Glocke** hat.
8. Schreiben Sie als Beispiel folgende kurze Sätze in das Dokument:
   Das Junge der Kuh heisst Kalb. Das Junge der Henne heisst Küken. Das Junge des Pferdes heisst Fohlen.
9. Formatieren Sie nun die Wörter **Kalb**, **Küken** und **Fohlen** mit der Formatvorlage mit der Bezeichnung **Lückentext Lehrer**. Dies ist eine der Zeichen-Formatvorlage, welche der Code generiert hat. Die zweite nennt sich übrigens *Lückentext Schüler*.
10. Klicken Sie nun die Schaltfläche mit der Glocke an. Dabei werden die Lösungsworte ausgeblendet, zurück bleibt jeweils eine unterstrichene Lücke.
11. Bei nochmaligem Klick auf das selbe Symbol, rastet der Knopf wieder ein und die Lösung erscheint wieder auf dem Bildschirm. Das Dokument kann in beiden Ansichten auf Wunsch ausgedruckt werden.

VBA-Code

Am anschaulichsten präsentiert sich die Lösung in **Word2000**. Der Unterschied zwischen Word 97 und Word2000 liegt darin, dass es in Word2000 möglich ist, andersfarbig zu unterstreichen. Im Einsatz kann also die Schrift auf *Weiss* gesetzt werden, wogegen der Unterstrich in einer anderen Farbe, als gut sichtbar bleibt.

Ich habe mich deshalb entschlossen, sowohl eine Lösung für Word2000, wie auch eine Lösung für Word 97 zu publizieren, trotz der Tatsache, dass sich der Code nur geringfügig unterscheidet. Sie benötigen lediglich eines der folgenden Code-Blöcke.

Word2000

Kopieren Sie den folgenden Code in ein neues Modul Ihrer **Normal.dot** und lassen Sie das Makro mit dem Namen **LueckentextEinrichten**.

```
Sub LueckentextEinrichten()
  Application.CustomizationContext = NormalTemplate
  Dim Knopf As CommandBarButton, FV As Style
  Set Knopf = CommandBars.FindControl(Tag:="SW20010203")
  If Not Knopf Is Nothing Then Knopf.Delete
  Set Knopf = CommandBars("Standard").Controls.Add(ID:=1)
  With Knopf
    .Tag = "SW20010203"
    .Caption = "Lückentext"
```

```
    .TooltipText = "Blendet die Lösung aus."
    .OnAction = "LueckentextProzess"
    .FaceId = 273
    .State = msoButtonDown
  End With
  flag = False
  For Each FV In ActiveDocument.Styles
    If FV.NameLocal = "Lückentext Lehrer" Then
      flag = True
      Exit For
    End If
  Next
  If flag = False Then
    Set FV = ActiveDocument.Styles.Add(Name:="Lückentext Lehrer", _
       Type:=wdStyleTypeCharacter)
    FV.Font.ColorIndex = wdGreen
    FV.Font.Underline = wdUnderlineSingle
    FV.Font.UnderlineColor = wdColorGreen
  End If
  flag = False
  For Each FV In ActiveDocument.Styles
    If FV.NameLocal = "Lückentext Schüler" Then
      flag = True
      Exit For
    End If
  Next
  If flag = False Then
    Set FV = ActiveDocument.Styles.Add(Name:="Lückentext Schüler", _
       Type:=wdStyleTypeCharacter)
    FV.Font.ColorIndex = wdWhite
    FV.Font.Underline = wdUnderlineSingle
    FV.Font.UnderlineColor = wdColorDarkRed
  End If
  Application.OrganizerCopy Source:=ActiveDocument.FullName, _
     Destination:=NormalTemplate.FullName, Name:="Lückentext Lehrer", _
        Object:=wdOrganizerObjectStyles
  Application.OrganizerCopy Source:=ActiveDocument.FullName, _
     Destination:=NormalTemplate.FullName, Name:="Lückentext Schüler", _
        Object:=wdOrganizerObjectStyles
  NormalTemplate.Save
End Sub

Private Sub LueckentextProzess()
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars.ActionControl
  If Knopf.State = msoButtonUp Then
    Knopf.State = msoButtonDown
    Knopf.TooltipText = "Blendet die Lösung aus."
    p1 = "Lückentext Schüler"
    p2 = "Lückentext Lehrer"
  Else
    Knopf.State = msoButtonUp
    Knopf.TooltipText = "Blendet die Lösung ein."
```

```
    p1 = "Lückentext Lehrer"
    p2 = "Lückentext Schüler"
  End If
  With ActiveDocument.Content.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Format = True
    .Text = ""
    .Style = p1
    .Replacement.Style = p2
    .Execute Replace:=wdReplaceAll
  End With
End Sub
```

(Das Lückentextmakro für *Word2000*. Die Version für Word 97 folgt gleich anschliessend.)

Word 97

Kopieren Sie den folgenden Code in ein neues Modul Ihrer **Normal.dot** und lassen Sie das Makro mit dem Namen **LueckentextEinrichten**.

```
Sub LueckentextEinrichten()
  Application.CustomizationContext = NormalTemplate
  Dim Knopf As CommandBarButton, FV As Style
  Set Knopf = CommandBars.FindControl(Tag:="SW20010203")
  If Not Knopf Is Nothing Then Knopf.Delete
  Set Knopf = CommandBars("Standard").Controls.Add(ID:=1)
  With Knopf
    .Tag = "SW20010203"
    .Caption = "Lückentext"
    .TooltipText = "Blendet die Lösung aus."
    .OnAction = "LueckentextProzess"
    .FaceId = 273
    .State = msoButtonDown
  End With
  flag = False
  For Each FV In ActiveDocument.Styles
    If FV.NameLocal = "Lückentext Lehrer" Then
      flag = True
      Exit For
    End If
  Next
  If flag = False Then
    Set FV = ActiveDocument.Styles.Add(Name:="Lückentext Lehrer", _
       Type:=wdStyleTypeCharacter)
    FV.Font.ColorIndex = wdGreen
    FV.Font.Underline = wdUnderlineSingle
  End If
  flag = False
  For Each FV In ActiveDocument.Styles
    If FV.NameLocal = "Lückentext Schüler" Then
      flag = True
      Exit For
    End If
  Next
  If flag = False Then
```

```
    Set FV = ActiveDocument.Styles.Add(Name:="Lückentext Schüler", _
       Type:=wdStyleTypeCharacter)
    FV.Font.ColorIndex = wdWhite
    FV.Font.Underline = wdUnderlineSingle
  End If
  Application.OrganizerCopy Source:=ActiveDocument.FullName, _
     Destination:=NormalTemplate.FullName, Name:="Lückentext Lehrer", _
        Object:=wdOrganizerObjectStyles
  Application.OrganizerCopy Source:=ActiveDocument.FullName, _
     Destination:=NormalTemplate.FullName, Name:="Lückentext Schüler", _
        Object:=wdOrganizerObjectStyles
  NormalTemplate.Save
End Sub

Private Sub LueckentextProzess()
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars.ActionControl
  If Knopf.State = msoButtonUp Then
    Knopf.State = msoButtonDown
    Knopf.TooltipText = "Blendet die Lösung ein."
    p1 = "Lückentext Schüler"
    p2 = "Lückentext Lehrer"
  Else
    Knopf.State = msoButtonUp
    Knopf.TooltipText = "Blendet die Lösung aus."
    p1 = "Lückentext Lehrer"
    p2 = "Lückentext Schüler"
  End If
  With ActiveDocument.Content.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Format = True
    .Text = ""
    .Style = p1
    .Replacement.Style = p2
    .Execute Replace:=wdReplaceAll
  End With
End Sub
```

(Das Lückentextmakro für *Word 97*. Die Version für Word2000 finden Sie weiter oben)

Anmerkungen

- Sollte Ihnen das Erscheinungsbild der Lücken (oder das der Lösung) im Text nicht behagen, können Sie die Formatvorlage mit dem Namen **Lückentext Schüler** bzw. **Lückentext Lehrer** Ihren Bedürfnissen anpassen.
- Um z.B. die Lücke zu vergrössern, können Sie über ***Format - Zeichen - Zeichenabstand*** die **Laufweite vergrössern**.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Luecken.htm

René, Probst, Februar 2001

Änderungen

- --- nil ---

Der Dialog Einfügen - Symbol... zeigt statt den Symbolen nur leere Quadrate an

◄ Zurück zum Hauptindex

Symptome

> Wenn Sie den Word-Dialog *Einfügen - Sonderzeichen...* (Word 97) bzw. *Einfügen - Symbol...* (Word2000) aufrufen und z.B. die Schrift Wingdings wählen, erscheinen in der Tabelle nur leere Vierecke statt den erwarteten Symbolen.

Ursache

Der Grafikteil des Rechners kann diesen, in Wirklichkeit sehr komplexen Dialog, nicht richtig anzeigen.

Vorabklärungen

Ich kenne eine handvoll von Word-Problemen, welche ziemlich eindeutig mit der Grafikkarte bzw. dessen Treiber in Zusammenhang zu bringen sind (Microsoft kennt angeblich Hunderte!). Um mehr Klarheit über den Ursprung des Problems zu erhalten, können Sie folgende Versuche durchführen:

1. Verringern Sie die Auflösung Ihrer Anzeige am Bildschirm. Gehen Sie dabei wie folgt vor:
   - START - Einstellungen - Systemsteuerung - Anzeige.
   - Wechseln Sie zum Reiter Einstellungen.
   - Fahren Sie den Hebel bei Bildschirmbereich zurück.
   - Falls die Symbole bei dieser niedrigeren Bildschirmauflösung angezeigt werden, sollten Sie sich nach einem neuen bzw. anderen Grafikkartentreiber umschauen.
   - Alternativ könnten Sie auf sehr ähnliche Weise die Farbtiefe Ihrer Anzeige verringern. Ich wette mal, dass zwischen High Color (damit können 2 hoch 16 also rund 65'0000 Farbabstufungen angezeigt werden) und True Color (damit können 2 hoch 24 also rund 16 Mio. Farbabstufungen angezeigt werden), für die meisten Anwendungen kein sichtbarer Unterschied besteht.
2. Falls Ihr Rechner unter Windows 9x fährt, starten Sie den PC im abgesicherten Modus. Der abgesicherte Modus fährt Ihren Rechner mit einer Minimalkonfiguration an Treibern hoch. Der Bildschirm wird unter diesen Konditionen ziemlich flackern. Sie erreichen dies wie folgt:
   - Sichern Sie Ihre Arbeit.
   - Führen Sie den Vorgang ***START - Beenden - Neu starten*** aus.
   - Wählen Sie die Option **Neu starten** und betätigen Sie die Schaltfläche ***Ok***.
   - Der Rechner fährt nun kurz herunter und startet anschliessend gleich wieder auf.
   - Achten Sie nun ganz genau auf die Meldungen, welche beim Hochfahren erscheinen.
   - Sowie der Meldungsfluss vom BIOS vorbei ist und bevor Windows startet, betätigen Sie die Taste **[Strg]** **Windows 98** bzw. die Taste **[F8]** **Windows 95** und halten diese gedrückt.
   - Der Rechner sollte jetzt, noch auf DOS-Ebene, ein Menü Anzeigen.
   - Wählen Sie den Punkt: **Abgesicherter Modus**.
   - Wenn der Rechner wieder betriebsbereit ist, überprüfen Sie die Anzeige der Symbole.
   - Falls die Symbole in diesem Modus korrekt angezeigt werden, sollten Sie sich nach einem neuen bzw. anderen Grafikkartentreiber umschauen.

3. Fahren Sie die Hardwarebeschleunigung Ihrer Grafikkarte zurück. Gehen Sie dabei wie folgt vor:
   - START - Einstellungen - Systemsteuerung - System.
   - Wechseln Sie zum Reiter Leistungsmerkmale.
   - Betätigen Sie die Schaltfläche Grafik....
   - Fahren Sie den Hebel bei Hardwarebeschleunigung auf 33% zurück.
   - Bestätigen Sie diese Aktion mit Ok (2x)
   - Starten Sie Ihren Rechner neu.
   - Wenn der Rechner wieder betriebsbereit ist, überprüfen Sie die Anzeige der Symbole.
   - Falls die Symbole bei dieser konservativen Einstellung der Grafikkarte korrekt angezeigt werden, können Sie entweder die Grafikkartenbeschleunigung stufenweise wieder hoch fahren oder sich nach einem neuen bzw. anderen Grafikkartentreiber umschauen.

Abhilfe

- Besorgen Sie sich einen neuen Treiber beim Hersteller der Grafikkarte. Meistens bietet dieser auf seiner Internet-Seite gratis Downloads für seine Produkte an. Falls Sie nicht wissen, über welche Grafikkarte Ihr Rechner verfügt, dann achten Sie beim Aufstarten des Rechners was diesbezüglich angezeigt wird. In der Regel identifiziert sich die Grafikkarte selbst. Es ist oft die aller erste Meldung, welche vom BIOS angezeigt wird. Falls Sie diese Meldung nicht erkennen können, weil diese zu schnell überschrieben wird, können Sie im richtigen Augenblick die **Pause**-Taste betätigen.

  Verrückt aber wahr: die Treiber, welche der Karte beim Kauf beiliegen sind oft bereits überholt!

  Manchmal ist auch nicht der neuste Treiber das Beste für Sie. Wenn Ihr System unter **Windows 98** läuft, sollten Sie auch einen Blick auf die Treiber werfen, welche von Microsoft ausgeliefert werden. Eigentlich wäre dies für Windows 98 Benutzer ein geeigneter Fall für einen sogenannten **Live-Update** Ihres Treibers. Nun zeigt aber die Erfahrung, dass diese Treiberdatenbank von Microsoft ein Nepp ist. Meines Wissens hat dort noch keiner etwas gefunden. Die Antwort ist immer die selbe: der optimale Treiber sei bereits installiert.

- Stellen Sie die Hardwarebeschleunigung Ihrer Grafikkarte konservativ ein, wie dies oben beschrieben wurde.

Links zu diesem Thema

Microsoft

Deutsch

► **Quadratische Felder statt Symbole im Dialogfeld Symbol einfügen**

Englisch

► **WD97: Squares Not Symbols in Insert Symbol Dialog Box**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Symbole.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Word vergisst die Skalierung von verknüpften Grafiken

◄ Zurück zum Hauptindex

Symptome

> In Word eingebettet Grafiken können skaliert (in der Grösse verändern) oder sogar verzerrt werden. Dies geschieht über den Word-Dialog *Grafik formatieren - Grösse.* Plötzlich, beim Öffnen des Word-Dokumentes, stellen Sie nun aber fest, dass diese Skalierung verloren ging und das Bild in der Originalgrösse angezeigt wird. Um so ärgerlich, wenn Sie viel Zeit und Mühe aufgewendet haben, Ihrem Dokument ein schönes Layout zu geben.

Abhilfe

- Sie sollten den **Office 97 Service Release 1** installieren, falls Sie dies noch nicht getan haben. Der SR-1 bringt hier Linderung, allerdings ist das Problem damit leider nicht immer vom Tisch, obwohl Microsoft dies behauptet.
- Um ganz sicher vor solchen unliebsamen Überraschungen zu sein, sollten Sie Grafiken ein **Korsett** verpassen. Sie können die Grafik in ein Textfeld oder in einen Positionsrahmen einbetten. Auch Zellen von Tabellen eigenen sich dazu. Achten Sie in Word 97 darauf, dass beim Einfügen der Grafik die Option *über den Text legen* **nicht** aktiviert ist. Nur so ist es möglich, eine Grafik in ein Textfeld, einen Positionsrahmen oder in eine Tabelle zu platzieren. Falls Sie dies missachten, landet die Grafik ausserhalb dieses Gefässes. Am besten eignen sich wohl **Positionsrahmen**.

Anmerkungen

- Vorausgesetzt Sie verpassen der Grafik kein Korsett verlieren Sie die Skalierung ganz bestimmt, wenn Sie die Option *Grafik im Dokument speichern* (diese Option erreichen Sie über ***Bearbeiten - Verknüpfungen***) ein- bzw. ausschalten. Dies geschieht auch dann, wenn das Objekt gesperrt wird, also die dynamische Aktualisierung des Objektes aus der Quelldatei bewusst unterbunden wird. Eine technische Notwendigkeit für dieses störende Verhalten sehe ich nicht und man konnte es mir auch nicht erklären; es ist nun mal so (*It's Microsoft!*). Solange Microsoft dieses Problem nicht richtig anerkennt bzw. falsch einschätzt, wird sich an diesem unerfreulichen Zustand wohl kaum was ändern.
- Anders als in früheren Word-Versionen, steht der Befehl ***Einfügen - Positionsrahmen*** auf den ersten Blick nicht mehr zur Verfügung. Der Befehl ist allerdings immer noch unterstützt und Sie können ihn auf einfache Weise reaktivieren. Wie Sie diesen Befehl reaktivieren können, erfahren Sie in einem der unten stehenden Links.

Links zu diesem Thema

Microsoft

Deutsch

► Verknüpfte Grafik behält Größeneigenschaften nicht bei

Englisch

► WD97: Linked Picture Doesn't Preserve Scaling Properties

Eigene Links

► Befehl "Positionsrahmen einfügen" reaktivieren

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/GrafikS.htm
René Probst, Oktober 1999

Wie kann ich obsolete Einträge zu Dateien im Arbeitsmenü oder in der Auflistung der zuletzt geöffneten Daten löschen?

◄ Zurück zum Hauptindex

Symptome

> 1. Gerne benutze ich die Einrichtung in Word, welche mir erlaubt, ein Menü mit Verknüpfungen zu all jenen Dateien zu führen, welche ich sehr oft bearbeite. Ab und zu lösche ich aber eine dieser Dateien oder ich nenne die Datei um. Unschön dabei ist jetzt, dass die Verknüpfung im Menü erhalten bleibt und nun verwaist ist.
>
> 2. Im Menü Datei finde ich eine Auflistung der zuletzt geöffneten Dateien, welches mir erlaubt sehr bequem auf diese Dateien zuzugreifen. Lösche ich nun aber eine dieser Dateien, bleibt die Verknüpfung in dieser Auflistung erhalten und ist nun verwaist.

Antwort

Um solche Einträge zu löschen, gehen Sie wie folgt vor:

1. Drücken Sie die Tastenkombination **[Strg + Alt + -]** (Bindestrich). Die Einfügemarke verwandelt sich dabei in einen dicken waagerechten Balken.
2. Öffnen Sie nun das Menü Ihrer Wahl und klicken Sie auf den Eintrag, welcher Sie los haben möchten.
3. Wiederholen Sie diesen Vorgang allenfalls für andere obsolete Einträge.
4. Möchten sie diesen Zustand wieder aufheben, drücken Sie die Taste **[Esc]**

Anmerkungen

- Sie können auf diese Art fast beliebige Menüeintrage löschen. Gehen Sie also mit einer gewissen Sorgfalt ans Werk.
- Falls Sie diesen Befehl aus irgend welchen Gründen auf eine andere Tastenkombination legen möchten (was Sie über *Extras - Anpassen - Tastatur... - Alle Befehle* erreichen), dann sollten sie wissen, dass sich dieser Befehl wie folgt nennt: *ExtrasAnpassenShortcutEntfernen*.
- Möchten Sie sämtliche Einträge in der Auflistung der zuletzt geöffneten Dateien in einem Rutsch löschen, dann deaktivieren Sie die Option ***Extras - Optionen - Allgemein - Liste der zuletzt geöffneten Dateien*** und reaktivieren Sie diese Option anschliessend wieder..
- Eine automatisierte Lösung für dieses Problem, finden Sie etwas weiter unten in diesem Skript.

Einrichten des Arbeitsmenüs

Sollte das Arbeitsmenü für Sie kein bekannter Begriff darstellen, dann wird es Sie interessieren, wie Sie es nutzen können:

1. Rufen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen*** auf.
2. Wechseln Sie in der Reiter **Befehle**.
3. Markieren Sie in der linken Liste *Kategorien:* den Eintrag **Eingebaute Menüs**.
4. Suchen Sie in der rechten Liste Kategorien: nach dem Eintrag **Arbeit** und ziehen Sie nun diesen Eintrag mit der Maus auf eine Symbolleiste Ihrer Wahl.

Anmerkungen

- Um anschliessend eine Datei dem Arbeitsmenü hinzuzufügen, öffnen Sie zuerst die gewünschte Datei und wählen Sie dann vom Word-Hauptmenü den Befehl ***Datei - Zum Arbeitsmenü hinzufügen***.
- Damit das Arbeitsmenü in der gesamtem Word-Anwendung zur Verfügung steht, machen Sie diese Anpassung am besten in der Standardvorlage mit dem Namen **Normal.dot**.

Zahl der Einträge, welcher unter der Auflistung der zuletzt geöffneten Dateien erscheint, beeinflussen

Die Anzahl der Einträge, welche in der genannten Auflistung erscheinen, lässt sich wie folgt beeinflussen. Beachten Sie, dass der Standardwert bei **4** und das Maximum bei **9** liegt.

1. Rufen Sie vom Word-Hauptmenü den Dialog ***Extras - Optionen*** auf.
2. Wechseln Sie in den Reiter **Allgemein**.
3. Passen Sie die Zahl unter **Liste der zuletzt geöffneten Dateien** Ihren Wünschen entsprechend an.

Synchronisation automatisiert erzwingen

Für die automatische Synchronisation der Einträge im Arbeitsmenü, habe ich leider keine konsistente Lösung gefunden.

Wenn Sie den Vorgang der Synchronisation betreffend der Auflistung der zuletzt geöffneten Dokumente automatisieren möchten, dann bietet sich als Aufhänger die Prozedur mit dem reservierten Namen **AutoExec** an. Den VBA-Code, welcher dann die Synchronisation an sich erledigt, finden Sie gleich anschliessend.

Um den Automatismus in Gang zu setzen, **öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** und kopieren den folgenden Code in ein neues Modul. Zusätzlich erstellen Sie im selben oder in einem anderen Modul der Normal.dot den Dreizeiler, welche ebenfalls unten abgedruckt ist.

```
Sub AutoExec()
  ListeDerZuletztGeöffnetenDateienSynchronisieren
End Sub
```

(Ruft die folgende Prozedur automatisch beim Starten von Word auf.)

```
Sub ListeDerZuletztGeöffnetenDateienSynchronisieren()
  Dim x As RecentFile
  For Each x In Application.RecentFiles
    chk = Dir(x.Path & "\" & x.Name)
    If chk = "" Then x.Delete
  Next x
End Sub
```

(Die oben abgebildete Prozedur synchronisiert die Einträge der zuletzt geöffneten Dateien im Menu Datei.)

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/AMenue.htm
René Probst, Juni 2000

Es stehen keine Seitenrahmen (Zierrahmen) zur Verfügung!

◄ Zurück zum Hauptindex

Problem

> **Ich arbeite mit Word 97 und habe kürzlich den Office 97 Service Release 1 installiert. Seither stehen mir diese putzigen Seitenrahmen unter *Datei - Seite einrichten... - Seitenlayout - Rahmen...* nicht mehr zur Verfügung. Ich kann da nichts anwählen, denn das Kombinationsfeld mit dem Namen *Effekte* enthält keine Einträge mehr.**

Ursache

Den Verantwortlichen bei Microsoft ist bei der Überführung der englischen Version des Service Releases 1 in die deutschsprachige Version eine Übersetzungsfehler unterlaufen, welcher bewirkt, dass dieses Feature faktisch unbrauchbar ist.

Lösung

Lassen Sie nach einem Verzeichnis mit dem Namen **Zierrahmen** suchen. Dabei handelt es sich um ein Unterverzeichnis des Office-Verzeichnisses. Nennen Sie diesen Ordner nach Rahmen um.

Anmerkung

- Dieses Problem beschränkt sich auf die deutschsprachige Ausgabe von Office 97 mit Service Release 1
- Um zu Prüfen, auf welchem Stand Ihre Office-Installation ist, starten Sie Word, rufen Sie vom Hauptmenü den Dialog ***? - Info*** auf, womit sich ein Dialog öffnet. Sollte dort im Titel nichts stehen oder aber *SR-2* sind Sie **nicht betroffen**.
- Office2000 Benutzer und Benutzerinnen sind nicht betroffen. In dieser Version heisst das entsprechende Verzeichnis übrigens *Borders*.

Links zu diesem Thema

Microsoft

Deutsch

► **Keine Seitenrahmen-Effekte bei Office 97-Workstationsetup**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/ZRahmen.htm

René Probst, Oktober 2000

Änderungen

- --- nil ---

Das Betätigen der Enter-Taste in einem konventionellen Formulartextfeld erzeugt einen Zeilenumbruch und verschiebt die Darstellung in meinem Formular

◄ Zurück zum Hauptindex

Symptome

> Betätigt der Benutzer die Eingabe-Taste in einem Formularfeld wird in meinem sorgfältig aufgebauten Formular eine zusätzliche Zeile eingefügt. Damit gerät das Layout durcheinander.

Erklärung

In früheren Versionen von Word hatte die Eingabe-Taste bei Formularfeldern eine andere Funktion, nämlich die selbe wie die **Tabulatorentaste**. Die Einfügemarke sprang also zum nächsten Eingabefeld im Formular. Zu Gunsten der Kompatibilität mit Formularen im Internet, wurde dieses Verhalten aber bewusst von Microsoft geändert.

Abhilfe

Helfen kann hier nur eine VBA-Makrolösung. Dabei sind zwei Lösungen in Betracht zu ziehen. Die Minimallösung stellt die Funktion der Eingabe-Taste innerhalb eines Formulartextfeldes ganz ab, die Maximalvariante ändert die Funktionalität der Eingabe-Taste im Umfeld mit Formularfeldern. Ziel der zweiten Lösung ist es, das Verhalten der Eingabe-Taste den früheren Versionen von Word anzugleichen. Diese Taste soll wieder als Tabulatorentaste fungieren, also bewirken, dass in das nächste Eingabefeld gesprungen wird. Die erste Variante verhindert zumindest, dass der Endbenutzer das Layout des Formulars durcheinander bringen kann.

Beide Lösungen nutzen einerseits die Möglichkeit von Word 97 oder Word2000 aus, einer Taste ein beliebiges Makro zuzuweisen. Anderseits ist es auch möglich, jedem Textformularfeld sowohl ein Entritts- so wie auch ein Austrittsmakro zuzuweisen. Kombiniert man die beide Optionen, gelangt man zu einem Mechanismus, welcher innerhalb eines Textformularfeldes speziell reagiert und ausserhalb dieser Felder wieder normal, also wie von Word üblicherweise vorgesehen.

Die Makros um das Verhalten der Eingabetaste im erklärten Sinne zu ändern, sehen wie folgt aus:

VBA-Code

Minimalvariante

```
Sub AutoNew()
  ActiveDocument.Protect Type:=wdAllowOnlyFormFields, NoReset:=True
  If Selection.Bookmarks.Count = 1 Then EnterOff
End Sub

Sub EnterOff()
  CustomizationContext = ActiveDocument.AttachedTemplate
  KeyBindings.Add KeyCategory:=wdKeyCategoryMacro, _
     Command:="Dummy", KeyCode:=vbKeyReturn
End Sub

Sub EnterOn()
  CustomizationContext = ActiveDocument.AttachedTemplate
  FindKey(BuildKeyCode(Arg1:=vbKeyReturn)).Clear
  ActiveDocument.AttachedTemplate.Save
End Sub

Sub Dummy()
End Sub
```

(Die Funktion der Eingabetaste wird in Formularfeldern unterbunden.)

Implementierung

Um diese Lösung in Ihr Projekt einzubringen, gehen Sie bitte wie folgt vor:

1. **Öffnen** Sie die Dokumentenvorlage auf dem das Formular basiert.
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
3. Wählen Sie dort vom Hauptmenü den Befehl ***Einfügen - Modul***.
4. Kopieren Sie den gesamten Code, also alle vier Prozesse in das leere Fenster.
5. Kehren Sie zum Word-Fenster zurück.
6. Falls das Formular geschützt ist, heben Sie den Schutz auf.
7. Führen Sie einen Rechtsklick auf das erste Formulartextfelder aus und wählen Sie im Kontextmenü ***Eigenschaften***.
8. Wählen Sie für das Feld ***Eintrittsmakro*** den Eintrag **EnterOff** aus.
9. Wählen Sie für das Feld ***Austrittsmakro*** den Eintrag **EnterOn** aus.
10. Wiederholen Sie Punkt 7 bis 9 für alle Formulartextfelder.
11. **Wichtig:** Schützen Sie das Dokument **nicht**. Dies wird der Code erledigen.
12. Speichern Sie die Dokumentenvorlage und testen Sie die Sache.

Die Schritte sieben bis neuen liessen sich auch dank einem zusätzlichen Makro automatisieren. Der Code für dieses Makro könnte dann wie folgt aussehen:

Feldeintritts- und Feldaustrittsmakro automatisiert setzen

```
Sub FormularfeldEreignisseSetzen()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Dim x As FormField
  If oDoc.ProtectionType > 0 Then oDoc.Unprotect
  For Each x In oDoc.FormFields
    If x.Type = wdFieldFormTextInput Then
      x.EntryMacro = "EnterOff"
      x.ExitMacro = "EnterOn"
    End If
  Next x
End Sub
```

(Alle Textformularfelder des Dokumentes erhalten ein Eintritts- sowie ein Austrittsmakro verpasst. Die Prozesse mit den Namen *EnterOff* und *EnterOn* müssen bereits in der Vorlage enthalten sein.)

Maximalvariante

```
Sub AutoNew()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  CustomizationContext = oDoc.AttachedTemplate
  KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyReturn), _
     KeyCategory:=wdKeyCategoryMacro, Command:="TabFunction"
  KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyShift, wdKeyReturn), _
     KeyCategory:=wdKeyCategoryMacro, Command:="TabFunction"
  oDoc.Protect Type:=wdAllowOnlyFormFields, NoReset:=True
  Templates(oDoc.AttachedTemplate.FullName).Save
End Sub

Sub AutoOpen()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  CustomizationContext = oDoc.AttachedTemplate
  KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyReturn), _
     KeyCategory:=wdKeyCategoryMacro, Command:="TabFunction"
  KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyShift, wdKeyReturn), _
     KeyCategory:=wdKeyCategoryMacro, Command:="TabFunction"
End Sub

Sub AutoClose()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  CustomizationContext = oDoc.AttachedTemplate
  FindKey(KeyCode:=BuildKeyCode(wdKeyReturn)).Disable
  FindKey(KeyCode:=BuildKeyCode(wdKeyShift, wdKeyReturn)).Disable
  Templates(oDoc.AttachedTemplate.FullName).Save
End Sub

Sub TabFunction()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  If oDoc.ProtectionType > 0 And _
   oDoc.Sections(Selection.Information(wdActiveEndSectionNumber)).ProtectedForForms _
      = True Then
    x = Selection.Bookmarks(1).Name
    If oDoc.FormFields(x).Name <> oDoc.FormFields(oDoc.FormFields.Count).Name Then
      oDoc.FormFields(x).Next.Select
    Else
      oDoc.FormFields(1).Select
    End If
  Else
    Selection.TypeText Text:=Chr(13)
  End If
End Sub
```

(Die Funktionsweise der Eingabetaste (und ebenfalls Umschalt + Eingabetaste) wird in Formularfeldern der der Tabulatorentaste gleichgesetzt. Die Namen der Prozesse Auto... sind zwingend.)

Implementierung

Um diese Lösung in Ihr Projekt einzubringen, gehen Sie bitte wie folgt vor:

1. **Öffnen** Sie die Dokumentenvorlage auf dem das Formular basiert.
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
3. Wählen Sie dort vom Hauptmenü den Befehl ***Einfügen - Modul***.
4. Kopieren Sie den gesamten Code, also alle drei Prozesse in das leere Fenster.
5. Kehren Sie zum Word-Fenster zurück.
6. **Wichtig:** Falls das Formular geschützt ist, heben Sie den Schutz auf. Die Vorlage muss also ungeschützt gespeichert werden.
7. Speichern Sie die Dokumentenvorlage und testen Sie die Sache.

Anmerkung

! Bedenken Sie, dass alle Varianten - auch die Variante, welche Microsoft im folgenden Skript vorschlägt, ihre Schwächen haben. Die Problematik liegt vor allem darin, dass sich die Lösungen kaum weiter ausreizen lassen, und dass es gerade bei einer nicht stabilen Anwendung wie MS Word es darstellt, kritisch ist, in das Verhalten der Oberfläche einzugreifen. Die Lösungen gehen davon aus, dass immer alle automatisierten Schritte auch wirklich stattfinden und dass die Synchronität der Schalterfunktion, nie gestört wird. Schmiert z.B. Word zu einem besonders ungeschickten Moment ab, dann müssen Sie die Dokumentvorlage, auf welchem das Formular beruht von Hand nachbessern. Sie sollten dann dafür sorgen, dass das Verhalten der Eingabe-Taste, wieder dem entspricht, wie es Word nun mal vorsieht. Sie erreichen dies über *Extras - Anpassen - Tastatur...*

Links zu diesem Thema

Microsoft

Deutsch

► **WD97: Codierung der ENTER-Taste: Cursor ins nächste Formularfeld**

Englisch

► **WD97: How to Code the ENTER Key to Move to Next Field in Protected Form**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Enter.htm

René Probst, Oktober 1999

Änderungen

- 21. April 2000 - VBA-Code für Maximalvariante angepasst. Auch [Umschalt + Enter] bewirkt nun den Tab-Effekt
- 27. Februar 2001 - VBA Code für beide Varianten verbessert
- 27. Februar 2001 - Bemerkung, dass sich diese Sache wohl kau tot testen lässt

Wie kann ich leere Absätze (Zeilen) aus einem Dokument entfernen?

Wie kann ich Text, bei dem jede Zeile mit einer Absatzmarke oder einer manuellen Zeilenumschaltung endet, in Fliesstext umwandeln?

◄ Zurück zum Hauptindex

Problem

> 1. Ich habe ein umfangreiches Word-Dokument geerbt. Leider enthält das Dokument unzählige leere Absatzmarken, da der Autor der Datei, statt den Abstand zwischen Absätzen richtig einzustellen, jeweils einfach die Eingabetaste betätigt hat. Wie kann ich diese leeren Absätze effizient entfernen?
>
> 2. Ich habe ein umfangreiches Word-Dokument geerbt. Leider wird in diesem Dokument, wie man das früher mit der Schreibmaschine getan hat, jede Zeile mit einer Absatzmarke bzw. einer manuellen Zeilenumschaltung abgeschlossen. Wie kann ich diesen Text in Fliesstext umwandeln?

Diese Seite

- ▼ Einleitung
- ▼ Terminologie
- ▼ Überzählige Absatzmarken (leere Absätze) eliminieren
- ▼ Überzählige manuelle Zeilenwechsel (leere Zeilen) eliminieren
- ▼ Jede Zeile endet mit einer Absatzmarke
- ▼ Jede Zeile endet mit einem manuellen Zeilenwechsel
- ▼ Wie kann ich auch nach anderen Formatierungszeichen suchen lassen?
- ▼ Gegenüberstellung Word 97 - Word2000 bezüglich der Formatierungszeichen
- ▼ Links zu diesem Thema

Einleitung

▲ Zurück zum Anfang

Sie können diese und ähnliche Übungen sehr effizient mit der sehr vielseitigen *Suchen*/Ersetzten-Funktion in MS Word erledigen. Bevor wir aber damit beginnen, möchte ich Begriffe, wie sie in MS Word (und nicht nur dort) in diesem Zusammenhang nun mal gelten, klar stellen.

Terminologie

▲ Zurück zum Anfang

Damit Sie die folgenden Erklärungen auch direkt in MS Word nachvollziehen können, sollten Sie sicherstellen, dass folgende Option in Word aktiviert ist: ***Extras - Optionen - Ansicht - Formatierungszeichen - Alle***.

Es gibt verschiedene Gründe, wieso der Text in einem Word-Dokument auf einer neuen Zeile beginnt bzw. weiterfährt. Im folgenden werden die möglichen Zeilenschaltungen aufgeführt. Jede Art der Zeilenschaltung hat ihren Sinn und sollte nicht missbräuchlich an Stelle der richtigen Methode verwendet werden. Dies ist im Interesse eines konsistenten Schriftstückes, ein Werk also, welches auch den Profis unter den Textverarbeitern Freude bereitet.

Fliesstext kommt ohne irgend eine Art von manueller Zeilenumschaltung aus. Wo nichts dagegen spricht, sollte man Fliesstext anwenden, also einfach ohne auf Umbrüche zu achten darauf los schreiben. Word wird dafür sorgen, dass der Text an einer geeigneten Stelle umbricht.

Neuer Absatz Dient dazu einen neuen Absatz zu beginnen. Diese Umbruchart sollte man immer dann verwendet, wenn ein neuer Gedankengang beginnt. Mehrere Absatzmarken hinter einander um den Abstand vom übrigen Text zu vergrössern oder um auf der Seite weiter nach unten zu kommen, ist in der elektronische Textverarbeitung verpönt. Dazu sollte man den Abstand zum nächsten Absatz über ***Format - Absatz*** einstellen. Um ein einheitliches Bild des Dokumentes zu erreichen, sollte man diese Einstellungen an die Formatvorlagen delegieren, also indirekt festlegen.

Manueller Zeilenumbruch. Diese Umbruchart sollte man, wenn überhaupt nur sehr spärlich einsetzen. In Word wird diese Art von Zeilenumbruch mit **[Umschalt + Eingabe]** erzwungen.

Soweit also zur Terminologie. Folgerichtig wird sich dieses Skript nun damit auseinandersetzen, wie überzählige Absatzmarken (also leere Absätze) und überzählige manuelle Zeilenumschaltungen (also leere Zeilen) effizient eliminiert werden können. Zudem wird auch gezeigt, wie Texte, welche am rechten Blattrand manuelle Zeilenumschaltungen bzw. Absatzmarken enthalten, in Fliesstext umgewandelt werden können.

Lösungen

Überzählige Absatzmarken (leere Absätze) eliminieren

▲ Zurück zum Anfang

Um mehrere auf sich folgende Absatzmarken zu eliminieren, gehen Sie bitte wie folgt vor:

Mittels Word-Dialog

1. Vom Word-Hauptmenü wählen Sie ***Bearbeiten - Ersetzen...***.
2. Im Textfeld mit der Bezeichnung *Suchen nach:* tippen Sie folgende Zeichenfolge ein: **^a^a Word 97** bzw. **^p^p Word2000**.
3. Im Textfeld mit der Bezeichnung *Ersetzen durch:* tippen Sie folgende Zeichenfolge ein: **^a Word 97** bzw. **^p Word2000**.
4. Betätigen Sie nun den Schaltknopf mit der Bezeichnung **Alle ersetzen**. Sie sollten diesen Knopf mehrfach drücken, solange halt bis keine Entsprechungen mehr gefunden werden.

Mittels Word-VBA

Im folgenden die VBA-Lösung, um überzählige Absatzmarken zu entfernen.

```
Sub ÜberzähligeAbsatzmarkenEntfernen()
  #If VBA6 Then
    x = "^p"
  #Else
    x = "^a"
  #End If
  ActiveDocument.Range(0, 0).Select
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = x & x
    .Replacement.Text = x
    Do While .Execute
      .Execute Replace:=wdReplaceAll
    Loop
  End With
  ActiveDocument.Range(0, 0).Select
End Sub
```

(Entfernt überzählige Absatzmarken in einem Word 97 oder Word2000 Dokument.)

Überzählige manuelle Zeilenwechsel (leere Zeilen) eliminieren

▲ Zurück zum Anfang

Um mehrere auf sich folgende manuelle Zeilenwechsel zu eliminieren, gehen Sie bitte wie folgt vor:

Mittels Word-Dialog

1. Vom Word-Hauptmenü wählen Sie ***Bearbeiten - Ersetzen...***.
2. Im Textfeld mit der Bezeichnung *Suchen nach:* tippen Sie folgende Zeichenfolge ein: **^z^z Word 97** bzw. **^l^l** (das sind kleine L's) **Word2000**.
3. Im Textfeld mit der Bezeichnung *Ersetzen durch:* tippen Sie folgende Zeichenfolge ein: **^z Word 97** bzw. **^l Word2000**.
4. Betätigen Sie nun den Schaltknopf mit der Bezeichnung **Alle ersetzen**. Sie sollten diesen Knopf mehrfach drücken, solange halt bis keine Entsprechungen mehr gefunden werden.

Mittels Word-VBA

Im folgenden nun die VBA-Lösung, um überzählige manuelle Zeilenwechsel zu entfernen.

```
Sub ÜberzähligeManuelleZeilenwechselEntfernen()
  #If VBA6 Then
    x = "^l"
  #Else
    x = "^z"
  #End If
  ActiveDocument.Range(0, 0).Select
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = x & x
    .Replacement.Text = x
    Do While .Execute
      .Execute Replace:=wdReplaceAll
    Loop
  End With
  ActiveDocument.Range(0, 0).Select
End Sub
```

(Entfernt überzählige manuelle Zeilenwechsel in Word 97 oder Word2000.)

Jede Zeile endet mit einer Absatzmarke

▲ Zurück zum Anfang

Sollten Sie ein Dokument geerbt haben, bei dem fast wie zu Schreibmaschinenzeiten jede Zeile mit einer Absatzmarke beendet wurde, können Sie diesen Text wie folgt in Fliesstext umwandeln:

Mittels Word-Dialog

1. Vom Word-Hauptmenü wählen Sie ***Bearbeiten - Ersetzen...***.
2. Im Textfeld mit der Bezeichnung *Suchen nach:* tippen Sie folgende Zeichenfolge ein: **^a Word 97** bzw. **^p Word2000**.
3. Im Textfeld mit der Bezeichnung *Ersetzen durch:* tippen Sie einen **Leerschlag** ein.
4. Betätigen Sie nun den Schaltknopf mit der Bezeichnung **Alle ersetzen**.

Mittels Word-VBA

Im folgenden nun die VBA-Lösung, um ein Text, bei dem jede Zeile rechts mit einer Absatzmarke begrenzt ist, in Fliesstext umzuwandeln.

```
Sub FliesstextStattAbsatzmarken()
  #If VBA6 Then
    x = "^p"
  #Else
    x = "^a"
  #End If
  ActiveDocument.Range(0, 0).Select
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = x
    .Replacement.Text = " "
    .Execute Replace:=wdReplaceAll
  End With
  ActiveDocument.Range(0, 0).Select
End Sub
```

(Wandelt einen Text, bei dem jede Zeile rechts mit einer Absatzmarke begrenzt ist, in Fliesstext um.)

Jede Zeile endet mit einem manuellen Zeilenwechsel

▲ Zurück zum Anfang

Sollten Sie ein Dokument geerbt haben, bei dem wie zu Schreibmaschinenzeiten jede Zeile mit einem manuellen Zeilenwechsel beendet wurde, können Sie diesen Text wie folgt in Fliesstext umwandeln:

Mittels Word-Dialog

1. Vom Word-Hauptmenü wählen Sie ***Bearbeiten - Ersetzen...***.
2. Im Textfeld mit der Bezeichnung *Suchen nach:* tippen Sie folgende Zeichenfolge ein: **^z** **Word 97** bzw. **^l** (das ist ein kleines L) **Word2000**.
3. Im Textfeld mit der Bezeichnung *Ersetzen durch:* tippen Sie einen **Leerschlag** ein.
4. Betätigen Sie nun den Schaltknopf mit der Bezeichnung **Alle ersetzen**.

Mittels Word-VBA

Im folgenden nun die VBA-Lösung, um ein Text, bei dem jede Zeile rechts mit einem manuellen Zeilenwechsel begrenzt ist, in Fliesstext umzuwandeln.

```
Sub FliesstextStattManuelle Zeilenwechsel()
  #If VBA6 Then
    x = "^l"
  #Else
    x = "^z"
  #End If
  ActiveDocument.Range(0, 0).Select
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = x
    .Replacement.Text = " "
    .Execute Replace:=wdReplaceAll
  End With
  ActiveDocument.Range(0, 0).Select
End Sub
```

(Wandelt einen Text in Word 97 oder Word2000, bei dem jede Zeile rechts mit einem manuellen Zeilenwechsel begrenzt ist, in Fliesstext um.)

Wie kann ich auch nach anderen Formatierungszeichen suchen lassen?

▲ Zurück zum Anfang

Sie können auch nach anderen Formatierungszeichen suchen lassen und diese allenfalls ersetzen, sei es nun durch ein druckbares Zeichen oder durch ein Formatierungszeichen. Gehen Sie wie folgt vor:

1. Vom Word-Hauptmenü wählen Sie ***Bearbeiten - Ersetzen...***.
2. Betätigen Sie die Schaltfläche **Erweitern**.
3. Im erweiterten Dialog klicken Sie nun auf **Sonstiges**.
4. Nun werden Ihnen zahlreiche Formatierungszeichen angeboten, welche Sie im Feld *Suchen nach:* bzw. *Ersetzen durch:* einsetzten können.

Gegenüberstellung Word 97 - Word2000 bezüglich der Formatierungszeichen

▲ Zurück zum Anfang

Die Bezeichnungen für Formatierungszeichen, welche übrigens in Word 97 noch «nicht druckbare Zeichen» genannt wurden, wurden auch in der deutschsprachigen Ausgabe von Office2000 der englischen Sprache angepasst.

Die nachfolgende Tabelle zeigt eine Gegenüberstellung von Word 97 und Word2000

Formatierungszeichen	Word 97	Word2000	Grund der Änderung
Absatzmarke	^a	^p	en. paragraph
Geschützter Bindestrich	^_	^~	öfters mal was Neues
Geschütztes Leerzeichen	^g	^s	en. space
Grafik	^r	^g	en. graphic (der Buchstabe g wurde jetzt frei)
Kommentarzeichen	^5	^a	en. annotation
Leerfläche	^l	^w	en. white
Manueller Zeilenwechsel	^z	^l	en. line

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Vom Umgang mit Zeilen und Seiten in MS Word

Microsoft Word MVP FAQ Site

► Remove all empty paragraphs from a document

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/LeereAbs.htm

René Probst, November 2000

Änderungen

- --- nil ---

Gegenüberstellung deutschsprachiger Feldnamen in Word 97 und englischsprachiger Feldnamen in Word2000

◄ Zurück zum Hauptindex

Problem

> Von früheren Word-Versionen bin ich es gewohnt, dass für Feldnamen deutschsprachige Begriffe zur Anwendung kommen. Seit ich auf Word2000 umgestellt habe, erscheinen diese Feldnamen nun aber in englischer Sprache. An welcher Schraube muss ich drehen, damit Word2000 die deutschsprachigen Begriffe verwendet?

Diese Seite

▼ Einleitung
▼ Tabelle der Feldnamen in Word 97 bzw. in Word2000
▼ Gegenüberstellung der Formatierungszeichen (nicht druckbare Zeichen) Word 97/Word2000
▼ Ungereimtheiten: International-deutsche Version versus deutsch-deutsche Version
▼ Probleme mit dem Datumsschalter bei der Übernahme von Word 97 Dokumenten nach Word2000
▼ Probleme mit der Feldfunktion «SUM» bei der Übernahme von Word 97 Dokumenten nach Word2000
▼ Felder mit VBA version- und sprachgrenzenüberschreitend einfügen
▼ Links zu diesem Thema

Einleitung

▲ Zurück zum Anfang

Sie können dieses Verhalten von Word2000 nicht beeinflussen. Word2000 (und wohl auch zukünftige Versionen) kennen nur noch die englischsprachigen Begriffe. Dabei ist zu beachten, dass gewisse **Zusätze** dieser Feldnamen gleichwohl noch deutschsprachig lauten müssen, so etwa in der Feldfunktion **Sum(Über)** (früher *Summe(Über)*). Andere Zusätze wiederum, müssen aber zwingend in englischer Sprache eingegeben wurden. So z.B. der allgemeine Schalter um ein Datum zu formatieren (früher *\@ "t. MMMM jjjj* jetzt unter Word2000 **\@ "d MMMM yyyy".** Dieser Sprachwirrwarr macht die Sache natürlich weder transparenter noch handlicher.

Die Umstellung - ging nicht ganz unerwartet - nicht ohne störende Nebengeräusche über die Bühne. So klappt z.B. die Umsetzung von benutzerdefinierten Datumsangaben nicht richtig, wenn ein Word 97 Dokument unter Word2000 geöffnet wird. Dieses Skript wird auch auf diese Problematik noch eingehen. Einige dieser Fehler werden übrigens mit dem Office2000 Service Release 2 von Microsoft bereinigt.

Ebenfalls geändert haben die Platzhalter für die *nichtdruckbaren Zeichen* (in Word2000 neu **Formatierungszeichen** genannt), welche z.B. im *Suchen/Ersetzen*-Dialog eingegeben werden können. Dieses Skript enthält auch dazu eine Tabelle, welche Sie über diese Änderung informiert.

Weiter geändert haben die Begriffe, welche Sie bei einem selektiven Druck des Dokumentes im Druck-Dialog eingeben müssen. Die Abkürzung für Seiten lautet neu «**p**» (englisch *page*), die jenige für Abschnitte «**s**» (englisch *section*).

Auch für Leute, welche in Word-VBA codieren könnte die Änderung von Konsequenz sein. Dann nämlich, wenn Sie mit Felder vom Typus **wdFieldEmpty** arbeiten. Diese Problematik wird später in diesem Skript noch näher erläutert.

Tabelle der Feldnamen in Word 97 bzw. in Word2000

▲ Zurück zum Anfang

Im folgenden finden Sie eine Gegenüberstellung der Feldnamen in englisch bzw. deutsch. In den linken zwei Spalten sind die Felder nach den deutschsprachigen Begriffen sortiert (wie es Word 97 vorschreibt) und in den zwei rechten Spalten sind die Felder nach den englischen Begriffen sortiert (wie Sie sie zwingend in Word2000 verwenden müssen).

Die Gross-/Kleinschreibung der Feldnamen ist übrigens irrelevant und dient ausschliesslich der besseren Lesbarkeit.

Word 97	Word2000	Word2000	Word 97
Deutsch	Englisch	Englisch	Deutsch
ABSCHNITT	Section	Advance	VERSETZEN
ABSCHNITTSEITEN	SectionPages	Ask	FRAGE
AKTUALDAT	Date	Author	AUTOR
ANGEBEN	Quote	AutoNum	AUTONR
ANZSEITEN	NumPages	AutoNumLgl	AUTONRDEZ
ANZWÖRTER	NumWords	AutoNumOut	AUTONRGLI
ANZZEICHEN	NumChars	AutoText	AUTOTEXT
AUTONR	AutoNum	AutoTextList	AUTOTEXTLISTE
AUTONRDEZ	AutoNumLgl	Comments	KOMMENTAR
AUTONRGLI	AutoNumOut	Compare	VERGLEICH
AUTOR	Author	CreateDate	ERSTELLDAT
AUTOTEXT	AutoText	DataBase	DATENBANK
AUTOTEXTLISTE	AutoTextList	Date	AKTUALDAT
BENUTZERADR	UserAddress	DocProperty	DOKEIGENSCHAFT
BENUTZERINITIALEN	UserInitials	DocVariable	DOKVARIABLE
BENUTZERNAME	UserName	Eq	FORMEL
BESTIMMEN	Set	FileName	DATEINAME
DATEIGRÖSSE	FileSize	FileSize	DATEIGRÖSSE
DATEINAME	FileName	Fill-In	EINGEBEN
DATENBANK	DataBase	GoToButton	GEHEZU
DATENSATZ	MergeRec	Hyperlink	HYPERLINK
DOKEIGENSCHAFT	DocProperty	If	WENN

Word 97	Word2000	Word2000	Word 97
Deutsch	Englisch	Englisch	Deutsch
DOKVARIABLE	DocVariable	IncludePicture	EINFÜGENGRAFIK
DOKVORLAGE	Template	IncludeText	EINFÜGENTEXT
DRUCK	Print	Index	INDEX
DRUCKDAT	PrintDate	Info	INFO
EINFÜGENGRAFIK	IncludePicture	Keywords	STICHWÖRTER
EINFÜGENTEXT	IncludeText	LastSavedBy	GESPEICHERTVON
EINGEBEN	Fill-In	Link	VERKNÜPFUNG
ERSTELLDAT	CreateDate	ListNum	LISTENNR
FORMEL	Eq	MacroButton	MAKROSCHALTFLÄCHE
FRAGE	Ask	MergeField	SERIENDRUCKFELD
FUSSENDNOTEREF	NoteRef	MergeRec	DATENSATZ
FVREF	StyleRef	MergeSeq	SERIENDRUCKSEQ
GEHEZU	GoToButton	Next	NÄCHSTER
GESPEICHERTVON	LastSavedBy	NextIf	NWENN
HYPERLINK	Hyperlink	NoteRef	FUSSENDNOTEREF
INDEX	Index	NumChars	ANZZEICHEN
INFO	Info	NumPages	ANZSEITEN
INHALT	TC	NumWords	ANZWÖRTER
KOMMENTAR	Comments	Page	SEITE
LISTENNR	ListNum	PageRef	SEITENREF
MAKROSCHALTFLÄCHE	MacroButton	Print	DRUCK
NÄCHSTER	Next	PrintDate	DRUCKDAT
NWENN	NextIf	Private	PRIVAT
PRIVAT	Private	Quote	ANGEBEN
RD	RD	RD	RD
REF	Ref	Ref	REF
SEITE	Page	RevNum	ÜBERARBEITUNGSNUMMER
SEITENREF	PageRef	SaveDate	SPEICHERDAT

Word 97	Word2000	Word2000	Word 97
Deutsch	Englisch	Englisch	Deutsch
SEQ	Seq	Section	ABSCHNITT
SERIENDRUCKFELD	MergeField	SectionPages	ABSCHNITTSEITEN
SERIENDRUCKSEQ	MergeSeq	Seq	SEQ
SONDZEICHEN	Symbol	Set	BESTIMMEN
SPEICHERDAT	SaveDate	SkipIf	ÜBERSPRINGEN
STICHWÖRTER	Keywords	StyleRef	FVREF
THEMA	Subject	Subject	THEMA
TITEL	Title	Symbol	SONDZEICHEN
ÜBERARBEITUNGSNUMMER	RevNum	TC	INHALT
ÜBERSPRINGEN	SkipIf	Template	DOKVORLAGE
VERGLEICH	Compare	Time	ZEIT
VERKNÜPFUNG	Link	Title	TITEL
VERSETZEN	Advance	TOC	INHALT
VERZEICHNIS	TOC	UserAddress	BENUTZERADR
WENN	If	UserInitials	BENUTZERINITIALEN
XE	XE	UserName	BENUTZERNAME
ZEIT	Time	XE	XE

Gegenüberstellung der Formatierungszeichen (nicht druckbare Zeichen) Word 97/Word2000

▲ Zurück zum Anfang

Die Bezeichnungen für **Formatierungszeichen**, welche übrigens in Word 97 noch *"nicht druckbare Zeichen"* genannt hiessen, wurden auch in der deutschsprachigen Ausgabe von Office2000 der englischen Sprache angepasst.

Die nachfolgende Tabelle zeigt dazu eine Gegenüberstellung von Word 97 und Word2000.

Formatierungszeichen	Word 97	Word2000	Grund der Änderung
Absatzmarke	^a	^p	en. paragraph
Geschützter Bindestrich	^_	^~	Das Tilde-Zeichen ist wohl vor allem auf einer englischen Tastatur schlecht erreichbar.
Geschütztes Leerzeichen	^g	^s	en. space
Grafik	^r	^g	en. graphic (der Buchstabe g wurde jetzt frei)
Kommentarzeichen	^5	^a	en. annotation
Leerfläche	^l	^w	en. white
Manueller Zeilenwechsel	^z	^l	en. line

Probleme und Ungereimtheiten

Die Umstellung bzw. Globalisierung bringt neben Vorteilen (die Vorzüge liegen vor allem im Bereich Automatisation und Programmierung) doch einiges an Ärger mit sich. Die neuralgischen Punkte werden im folgenden aufgeführt und erläutert.

International-deutsche Version versus deutsch-deutsche Version

▲ Zurück zum Anfang

Es ist eine Tatsache, dass Microsoft im deutschsprachigen Raum zwei Versionen von Office2000 vertreibt. Da wäre zum einen die **deutsch-deutsche** Version, welche weitaus beim Kunden die Verbreitteste ist. Bei dieser Version kommt es an einigen Stellen zu einem Durcheinander, muss doch der Feldbegriff an sich englisch lauten, die ergänzenden Parameter allerdings weiterhin in deutsch. Dies ist z.B. bei folgenden Ausdrücken der Fall:

- ! **Sum(Über)** statt Sum(Above)
- ! **Sum(Links)** statt Sum(Left)
- ! **Sum(Rechts)** statt Sum(Right)

☹ Trotz mehrmaliger Rückfrage konnte mir bisher niemand die recht simple Frage beantworten, wie sich diese unterschiedlichen Ausgaben effizient unterscheiden lassen. Daraus folgt auch, dass ich leider nicht weiss, ob bzw. wie eine Unterscheidung mittels VBA möglich ist, um wenigstens in einem Stück Code folgerichtig darauf reagieren zu können.

Probleme mit dem Datumsschalter bei der Übernahme von Word 97 Dokumenten nach Word2000

▲ Zurück zum Anfang

Wie schon angedeutet, muss der *allgemeine Schalter* um eine Feld vom Format *Datum* zu formatieren in Word2000 in englischer Sprache codiert werden. Konkret heisst dies, für den Platzhalter *Tag* muss neu **d** statt *t* und für den Platzhalter *Jahr* **y** statt *j* codiert werden. Neu muss es also z.B. **t. MMMM yyyy** lauten.

Werden nun aber Dokumente, welche in Word 97 geschrieben wurden in Word2000 geöffnet und damit in der Regel automatisch konvertiert, können diese Feldschalter nicht richtig aufgelöst werden und es kommt Unsinn dabei heraus.

☛ Dieses Problem wird von Microsoft mit dem *Office2000 Service Pack 2* bereinigt. Voraussetzung zur Installation dieses *Patches* ist der *Office2000 Service Release 1*. Sie finden am Ende dieses Skriptes die Links zu diesen Flicks.

Probleme mit der Feldfunktion «*SUM*» bei der Übernahme von Word 97 Dokumenten nach Word2000

▲ Zurück zum Anfang

Wird ein mit Word 97 erstelltes Dokument unter Word2000 geöffnet und dies unter der Annahme, dass dieses Dokument Feldfunktionen im Sinne von **Summe(Über)** enthält, dann wird zwar das Schlüsselwort *Summe* richtigerweise nach *Sum* übersetzt. Der Zusatz *Über* dagegen wird nicht übersetzt. Dies führt bei der international-deutschen Version von MS Word dazu, dass die Summe nicht ermittelt werden kann. Viel mehr erscheint an deren Stelle ein Syntaxfehler.

☛ Dieses Problem wird von Microsoft mit dem *Office2000 Service Pack 2* bereinigt. Voraussetzung zur Installation dieses Patches ist der *Office2000 Service Release 1*. Sie finden am Ende dieses Skriptes die Links zu diesen Flicks.

Felder mit VBA version- und sprachgrenzenüberschreitend einfügen

▲ Zurück zum Anfang

Es stellst sich die Frage, wie man vorgehen soll, wenn es darum geht, Felder mit VBA einzufügen, wobei dieser Code möglichst auch unter anderssprachigen Word-Ausgaben funktionieren sollte.

In den meisten Fällen - leider aber nicht in allen - ist das machbar. Sie sollten es sich abgewöhnen, den Feldtype `wdFieldEmpty` zu verwenden und dann den Text für den Feldcode in Klartext einzugeben. Damit schaffen Sie sich nur Kompatibilitätsprobleme.

Im folgenden Beispiel wird demonstriert, was der Makro-Recorder für einen Code generiert, wenn der Vorgang ***Einfügen - Feld...*** aufgezeichnet wird. Dieser generierte Code ist leider **sprachenabhängig**. Wenn Sie diesen in Word2000 erstellten Code z.B. in der deutschsprachigen Version von Word 97 abspielen, dann wird ein Feldcode geliefert, welche einen Syntaxfehler auslöst, statt das Erstelldatum anzuzeigen.

```
Selection.Fields.Add Range:=Selection.Range, Type:=wdFieldEmpty, Text:= _
    "CREATEDATE ", PreserveFormatting:=True
```

(Der Makro-Recorder zeichnet leider ein sprachenabhängigen Code auf...)

Sie können funktionell das gleiche selbst aber geschickter codieren, womit der Feldcode auch in anderen Sprachausgaben von Word und sowohl in Word 97 wie auch in Word2000 funktioniert. Das geht dann wie folgt:

```
Sub FeldSprachenunabhaengigEinfuegen()
  Selection.Fields.Add Range:=Selection.Range, Type:=wdFieldCreateDate, _
     Preserveformatting:=True
End Sub
```

(...dieser Code dagegen fügt einen versionen- und sprachenunabhängigen Code ein.)

Die Namen aller Feldkonstanten finden Sie, wenn Sie im **VBA-Editor** die Taste **[F2]** drücken, um den Objektkatalog anzuzeigen und dann nach dem Begriff `wdFieldType` suchen lassen.

Dümmer wird es, wenn Sie neben dem Schlüsselwort für das Feld auch noch frei wählbare Schalter angeben müssen, wie dies z.B. zur Formatierung eines Datums nötig ist. Möchten Sie dass die folgende Aufgabenstellung sowohl in der deutschsprachigen Ausgabe von Word 97, wie auch in jeder beliebigen Ausgabe von Word2000 funktioniert, dann kommen Sie an einer Abfrage in Ihrem VBA-Code nicht herum.

In den folgenden Beispielen wird eine Feld vom Format *Datum* in das Dokument eingefügt, wobei das Erscheinungsbild des Datums mit einem allgemeinen Schalter noch nachformatiert werden soll. Es stellt sich nun die lästige Frage der Kompatibilität. Dabei gilt folgendes:

- Das erste Beispiel wird ausschliesslich in der deutschsprachigen Ausgabe von Word 97 das Datum richtig anzeigen. Alle anderen Versionen bzw. Ausgaben von MS Word, werden den Feldcode mit sehr hoher Wahrscheinlichkeit mit einem Syntaxfehler quittieren. Wahrscheinlichkeit deshalb, weil es denkbar ist, dass es in einer anderen Sprache zufälligerweise deckungsgleich ist.
- Das zweite Beispiel wird in allen Ausgaben von Word2000 und auch in der englischsprachigen Ausgabe von Word 97 das Datum richtig anzeigen. Nicht-englischsprachige Ausgaben von Word 97 werden den Feldcode dagegen mit hoher Wahrscheinlichkeit mit einem Syntaxfehler quittieren.
- Das dritte und letzte Beispiel arbeitet mit einer Abfrage. Es sollte für alle Ausgaben von Word2000 sowie auch für die deutschsprachige Ausgabe von Word 97 den richtigen Feldcode erstellen.

```
Sub Beispiel1()
  Q = Chr(34)
  ActiveDocument.Fields.Add Range:=Selection.Range, _
     Type:=wdFieldCreateDate, Text:="\@" & Q & "t. MMMM. jjjj" & Q
  ActiveDocument.Fields.Update
End Sub
```

(Generiert einen Feldcode, welcher ausschliesslich in der deutschsprachigen Ausgabe von Word 97 richtig funktionieren wird.)

```
Sub Beispiel2()
  Q = Chr(34)
  ActiveDocument.Fields.Add Range:=Selection.Range, _
     Type:=wdFieldCreateDate, Text:="\@" & Q & "d. MMMM. yyyy" & Q
  ActiveDocument.Fields.Update
End Sub
```

(Generiert einen Feldcode, welcher in allen Sprachausgaben von Word2000 und in der englischsprachigen Ausgabe von Word 97 richtig funktioniert.)

```
Sub Beispiel3()
  Q = Chr(34)
  If Application.Version = "8.0" Then
    Schalter = "t. MMMM jjjj"
  Else
    Schalter = "d. MMMM yyyy"
  End If
  v = Val(Application.Version)
  ActiveDocument.Fields.Add Range:=Selection.Range, _
     Type:=wdFieldCreateDate, Text:="\@" & Q & Schalter & Q
  ActiveDocument.Fields.Update
End Sub
```

(Die wohl beste Lösung, welche mit probaten Mitteln erreichbar ist. Hier wird eine Feldfunktion generiert, welche sowohl in der deutschsprachigen Ausgabe von Word 97 wie auch in *allen* Word2000 Ausgaben funktioniert.)

Falls Sie sich neben Deutsch und Englisch mit weiteren Sprachen beschäftigen dürfen, können Sie analog vorgehen. Der Befehl, welcher darüber Aufschluss gibt, um welche Sprachausgabe von MS Word es sich handelt lautet: `Application.Language`. Dieser Ausdruck gibt eine vierstellige Zahl zurück, welche Sie wiederum im **Objektkatalog** (Taste **[F2]** im VBA-Editor) unter `msoLanguageID` nachschlagen können. Der Code für **Deutsch (Deutschland)** lautet übrigens **1031**.

Links zu diesem Thema

▲ Zurück zum Anfang

Microsoft

Englisch

► WD2000: Date/Time Form Fields Not Converted Correctly from Word 97
► WD2000: Syntax Error When You Update SUM Fields in Word 97 Document

Deutsch

► Office2000-Update: Service Pack 2 (SP-2)
► Office2000 Service Release 1a (SR-1a)

Eigene Links

► Wo kann ich den Office2000 Service-Release 1 beziehen?

Regionales Rechenzentrum der Universität Hamburg

► Verbalhornung (Eine gute Übersicht über die Änderungen in Word2000)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Felder.htm

René Probst, Januar 2001

Änderungen

- --- nil ---

Arbeiten mit dem Range-Objekt in MS Word

◄ Zurück zum Hauptindex

Frage

> Welche Bedeutung hat das Range-Objekt in Word-VBA? - Wie kann ich diese Eigenschaft bzw. dieses Objekt einsetzen, um meinen Code zu optimieren?

Diese Seite

- ▼ Was ist ein *Range*?
- ▼ Wieso ist die Hemmschwelle beim Arbeiten mit Ranges so gross?
- ▼ Vorteile von Range-Objekten (oder das flexibelste Word-Objekt überhaupt)
- ▼ Verzeichnis der hier präsentierten VBA-Lösungen bzw. Lösungsansätze
- ▼ Tabelle der Word-Objekte, welche über eine Range-Eigenschaft verfügen
- ▼ Ranges ansprechen
- ▼ Tabelle der Dokumententeile (StoryTypes) in MS Word
- ▼ Range-Objekte bilden
- ▼ Arbeiten mit der Eigenschaft FormattedText
- ▼ Arbeiten mit Textmarken
- ▼ Arbeiten mit Zeilen und Seiten
- ▼ Links zu diesem Thema

Was ist ein Range?

▲ Zurück zum Anfang

Es ist wirklich sehr einfach, den Begriff *Range* zu definieren: Ein Range stellt einen zusammenhängender Bereich in einem Word-Dokument dar. Der *Range* zeichnet sich durch einen Start und ein Ende aus (in Bezug auf den Dokumentteil, also z.B. Hauptteil des Dokumentes). Als wichtiger Spezialfall kann das Ende auch dem Start entsprechen. In diesem Fall wird ein Position und nicht ein Bereich im Dokument angesprochen. Genau dies ist z.B. der Fall, wenn der Range des Selection-Objektes abgefragt wird und es ist nichts markiert, bzw. die Markierung ist nicht erweitert.

Will man ein Word-Dokument *partout* so sehen, wie es sich auf dem Bildschirm präsentiert, was ich bei der Programmierung nicht gerade empfehle, dann sollte man sich bewusst sein, dass ein Bereich immer zusammenhängend sein muss und zwar in Leserichtung gesehen. Somit stellt eine Tabellenspalte kein einzelnen *Range* dar, was sich prompt auswirkt, verfügt doch das Objekt *Column* über keine *Range*-Eigenschaft. Sie können dieser Problematik aber recht geschickt ausweichen, wenn Sie eine Spalte als Sammlung von *Cell*-Objekten ansehen. Zellen verfügen über eine *Range*-Eigenschaft und sie können die Sammlung dieser Zellen in einer Schleife abarbeiten. Versinnbildlichen Sie sich das Word-Dokument am besten als eine lange Schnur, welche über den Bildschirm gelegt wird. Am Ende der Zeile, also am rechten Blattrand, wird die Schnur, welche Sie als ganzes sehen sollten umgebrochen. Stellt sich eine Grafik in den Weg, dann macht die Schnur halt etwas früher kehrt, es sei denn es wird eine spezielle Umbruchart gewählt, wie z.B. hinter den Text. Der Benutzer kann - das wissen wir - einen Zeilenumbruch auch erzwingen. Das ändert aber überhaupt nicht an der Tatsache, dass die Zeichenfolge "*xyz*" immer noch den selben Abstand gemessen vom Anfang der Schnur hat. Der Offset bleibt also immer konstant, gleichgültig davon, was diese Schnur auf dem Bildschirm für Kapriolen vollführt. Um präzise zu sein, sei noch erwähnt, dass ein Zeilenumbruch oder "neuer Absatz" den Platz eines Zeichens beansprucht auf dieser Schnur.

Die Kunst ein *Range*-Objekt anzusprechen bzw. zu erstellen besteht letztlich also darin, mit Word-VBA zu definieren, wo was anfängt und wo was aufhört (immer noch in Bezug auf den Dokumentteil, also z.B. Hauptteil des Dokumentes). Ein *Range* kann so klein sein, wie die Einfügemarke oder aber so gross, wie der ganze Hauptteil des Dokuments. Sie können dieser Kunst auf verschiedene Arten frönen - explizit, implizit oder gemischt:

- Ich will die Zeichen 10 - 20 im Hauptteil des Dokuments.
```
Set oRange = ActiveDocument.Range(Start:=9, End:=19)
```

- Ich will den Bereich des 2. Absatzes, allerdings ohne das letzte Zeichen (was die Absatzmarke darstellt).
```
Set oRange = _
   ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(2).Range.Start, _
      End:=ActiveDocument.Paragraphs(2).Range.End - 1)
```

- Ich will den Bereich, welcher mit dem Anfang des Absatzes 3 beginnt und mit dem Ende des Absatzes 6 endet.
```
Set oRange =
   ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(3).Range.Start, _
      End:=ActiveDocument.Paragraphs(6).Range.End)
```

- Ich will den Bereich, welche die 5. Tabelle im Dokument belegt
```
Set oRange = ActiveDocument.Tables(5).Range
```

Aus dieser Definition lässt sich ableiten, dass ein *Range*-Objekt sehr wenig Speicherplatz beansprucht. In der Tat, es wird bei der Bildung eines *Range*-Objektes überhaupt nichts dupliziert. Man kann mittels gültigem VBA-Code ein gedachtes *Range*-Objekt ansprechen oder man kann wirklich eines bilden, um später darauf zurück zu kommen. Auch im letzteren Fall handelt es sich aber ausschliesslich um eine Start/End-Definition (also einem Verweis auf den Text) und nicht um Daten, welche geladen und somit im Arbeitsspeicher dupliziert würden. Da ein *Range* ausschliesslich einen Verweis darstellt, stellt sich auch die Frage nicht, ob die Formatierung des Bereiches erhalten bleibt. Ein *Range*-Objekt kennt beide Eigenschaften, sowohl die Eigenschaft Text, wie auch die Eigenschaft *FormattedText*. Im ersten Fall ist der pure Text gemeint, im letzteren Fall der formatierte Text.

Jedes Zeichen (und ähnliches, wie z.B. Steuercode) gilt als eine Position. Der Offset (also die Zählung) beginnt bei 0. Das 10. Zeichen im Dokument hat also den Startbereich (`Range.Start`) 9.

Es ist nicht die Meinung, dass Sie nun beginnen die Zeichen im Dokument abzuzählen, um gedanklich zu einer bestimmten Position zu gelangen. Vielmehr weisen zahlreiche Objekte, welche sich in einem Word-Dokument tummeln, einen Start- bzw. Endbereich innerhalb des Dokumentes auf. Die kleinste Einheit welche Sie allenfalls in der Praxis mal abzählen ist der Absatz Orientieren Sie sich also an diesen Objekten. Machen Sie reichlich Gebrauch von Textmarken, denn auch Textmarken verfügen über eine *Range*-Eigenschaft, auf welche Sie sich im VBA-Code berufen können. Damit fällt das Abzählen dann ganz weg, da Textmarken einen eindeutigen Namen besitzen. Innerhalb des Bereichen eines Objektes können Unterobjekte auftauchen, welche ihrerseits wieder über einen Range verfügen.

Als Beispiel werfen wir einen Blick auf eine Word-Tabelle. Diese Tabelle stellt ein Bereich innerhalb des Dokumentes dar. Die Tabelle hat also einen Start- und Endpunkt. Innerhalb dieser Tabelle gibt es Zellen, welche wieder einen Bereich mit Start- und Endpunk aufweisen. Diese Positionen bzw. Bereiche lassen sich als Orientierungspunkte glänzen verwenden. Vor allem wird es möglich, z.B. eine solche Zelle einzulesen bzw. zu verändern, ohne dass Sie zuerst zur Tabelle und dort dann zur richtigen Zelle hüpfen müssten. Damit können Sie - quasi aus der Ferne - Elemente bearbeiten, ohne im Dokument zu navigieren.

Es ist zu bedenken, dass Word-VBA ein Dokument nicht als eine Einheit ansieht. Vielmehr kann ein Word-Dokument bis zu **elf Typen** von Teilen aufweisen. Davon ist der Hauptteil der bekannteste. Sie sprechen ihn in der Regel mit `ActiveDocument` an. Es gibt immer genau ein Hauptteil im Dokument (1:1 Relation). Die Teile nennen sich in der Fachsprache ***StoryRanges***. Manche dieser Typen können beliebig oft im Dokument vorkommen. Textfelder bilden dafür ein Beispiel (0 - n Relation, wobei Textfelder sogar noch verschachtelt werden können). Andere Typen wiederum kommen nicht oder genau einmal vor, wie z.B. der Endnotenbereich (0 oder 1 Relation). In jedem dieser *StoryRanges* beginnt die Zählung (Offset) wieder bei 0.

Sie finden gegen Ende dieses Skriptes eine Tabelle, mit den elf *StoryTypes*, welche das Word-Objekt-Modell definiert. Sie erfahren in diesem Dokument auch, wie die verschiedenen *StoryRanges* anzusprechen sind und wie diese allenfalls sequentiell (d.h. der Reihe nach ohne genau zu wissen, welche im Dokument tatsächlich vorhanden sind) abgearbeitet werden können. Das ist wichtig, den der Befehl `ActiveDocument.Fields.Update` z.B. wirkt sich ausschliesslich auf den Hauptteil des Dokumentes aus. Kopf- und Fusszeilen u.a. sind darin nicht eingeschlossen.

Es ist äusserst informativ, ja fast erleuchtend, die Architektur eines Word-Dokumentes im Bezug auf diese *StoryRanges* kennen zu lernen. Wichtig ist auch zu wissen, in welcher Anhängigkeit und Relation (Verhältnis) diese einzelnen Teilen zu einander stehen. So gibt es z.B. nicht einen Kopfzeilenbereich pro Dokument; nein es gibt deren drei für jeden Abschnitt (*section*) des Dokumentes!

Denken Sie abstrakt! Denken Sie in Strukturen und Architekturen! Zusammenhängend ist nicht zwingend das, was auf dem Bildschirm bzw. auf dem Ausdruck hinter einander folgt. Der Fussnotenbereich z.B. am Ende einer Seite liegt auf einer ganz anderen Ebene, als der Text, welcher den Fussnoten rein optisch vorausgeht. Die innere Struktur einer Word-Datei, folgt also in vielen Belangen nicht dem optischen Eindruck, den der Leser des Dokumentes erhält.

Wieso ist die Hemmschwelle beim Arbeiten mit Ranges so gross?

▲ Zurück zum Anfang

Wieso das *Range*-Objekt in MS Word bei vielen Leuten eine Art Mysterium darstellt, kann ich nur teilweise erklären. Aber auch ich habe mich anfänglich schwer getan dieses Hilfsmittel gekonnt einzusetzen. Bald habe ich aber zu mindest erahnt, dass sich hinter diesem Objekt ein grosses Potential verbirgt. Heute könnte ich wohl das *Range*-Objekt in VBA-Word nicht mehr missen. Ich arbeite mit Vorliebe damit.

Einige Gründe, wieso der Einstieg tatsächlich nicht einfach ist, lassen sich allerdings schon ausmachen, wie die folgenden Liste aufzeigen soll.

- Die Aufgabenstellung muss abstrakter formuliert werden, als wenn man mit dem *Selection*-Objekt arbeitet.
- Der Makro-Rekorder arbeitet nicht mit *Range*-Objekten, sondern zeichnet das Navigieren auf. Damit kann der Code, welcher vom Rekorder aufgezeichnet wird, nicht als Referenz hinhalten.
- Sowohl Logik wie auch Syntax ist bei *Range*-Objekten oft etwas komplexer als beim *Selection*-Objekt. Dafür werden in der Regel aber weniger Code-Zeilen gebraucht.

Vorteile von Range-Objekten (oder das flexibelste Word-Objekt überhaupt)

▲ Zurück zum Anfang

Das Arbeiten mit *Range*-Objekten hat zahlreiche Vorteile, wogegen kaum etwas gegen ihren Gebrauch spricht (ausser vielleicht die Gewohnheit). In der folgenden Liste werden die Vorteile von *Range*-Objekten gegenüber des *Selection*-Objektes aufgeführt.

- ☺ Sie können beliebig viele *Range*-Objekte ansprechen bzw. festlegen. Dazu ist es völlig unnötig im Dokument herum zu hüpfen. Das *Selection*-Objekt dagegen ist genau nur einmal vorhanden. Um weitere Teile einzulesen oder zu bearbeiten, müssen Sie die Einfügemarke an einen anderen Ort führen, also im Dokument herumhüpfen, will heissen im Dokument navigieren. Das ist genau dass, was Sie beim Eintippen zwangsläufig in der Praxis tun und was der Makro-Recorder bei Bedarf aufzeichnet. Der Code, welcher daraus resultiert ist allerdings oft weit vom Optimum entfernt.

- ☺ *Range*-Objekte machen das Navigieren im Dokument überflüssig. Das rumhopsen im Dokument ist unperformand, lässt während des Ablaufes eines Makros den Bildschirm nervös flackern und ist spätestens dann störungsanfällig oder sogar unmöglich, wenn MS Word ferngesteuert wird, allenfalls gar kein sichtbares Word-Fenster vorhanden ist, also Word im Hintergrund gestartet ist und von einer externen Anwendung gesteuert wird..

- ☺ Sie können Dokumententeile, ohne die Ansicht zu wechseln (z.B. Kopf-/und Fusszeilenbereiche) mit *Range*-Objekten jederzeit direkt bearbeiten. Schliesslich ist ja eine Datei im Arbeitsspeicher geladen und nicht nur eine Sequenz der Datei, welche zufälligerweise gerade am Bildschirm angezeigt wird.

Vielleicht kann man es auch so ausdrücken. Navigieren, dann dort was einlesen oder verändern, dann zur nächsten fraglichen Stelle navigieren dort wieder etwas anstellen, das ist Skript-Sprache. Sie könnten dies vermeiden, wenn Sie Range-Objekte einsetzen. Ein solcher Ansatz käme dann dem Vorgang des Programmierens schon viel, viel näher.

Lösen Sie sich doch ganz einfach von der Vorstellung, dass es sich bei einem Word-Dokument um ein Ding handelt, welches in Zeilen und in Seiten unterteilt ist. Diese Elemente mögen ja beim Druck wichtig sein. Gerade bei Fliesstext - und das ist die Stärke von MS Word - sind dies aber keine Strukturelemente.

Sehen Sie in einem Word-Dokument viel mehr ein Ding, welches im Arbeitsspeicher als reine Zeichenfolge vorliegt. Dank dem *Range*-Objekt können Sie einen Bereich in dieser Zeichenfolge ansteuern und einlesen bzw. verarbeiten. Dafür muss nicht auf dem Bildschirm herum gehopst werden.

Möchten Sie an einer bestimmten Stelle in dieser Zeichenfolge etwas einfügen, dann adressieren Sie ganz einfach einen *Range*, bzw. bilden Sie sich einen auf den Sie sich beziehen können und verwenden Sie anschliessend die *InsertBefore*- bzw. die *InsertAfter*-Methode.

Lösen Sie sich auch von der Vorstellung, welche Sie z.B. von Excel mit sich bringen. Wenn Sie eine neue Excel-Arbeitsmappe erstellen, erscheint dort ein Tabellenblatt mit einem vorgegebenen Raster, viele Zellen, welche zu einer Spalte und gleichzeitig zu einer Zeile gehören.

Sie können nun in Excel hingehen und die Zelle *C12* abfragen oder etwas rein schreiben. Die Abfrage wird zwar bei einer leeren Tabelle einen leere Zeichenfolge zurückgeben aber die Zelle ist da und adressierbar.

Anders in MS Word. Wenn Sie ein neues Word-Dokument erstellen, dann ist da wirklich gar nichts ausser einer einzelnen einsamen Absatzmarke. So sieht es nicht nur für den Benutzer aus, es ist Word-intern auch wirklich so. Im Gegensatz zu der Zelle *C12* auf einem Excel-Tabellenblatt ist die "Ortsangabe" *Paragraph* 3, Zeile 4, Position *12* also rein hypothetisch, will heissen vielleicht gibt es ja erst einen Absatz im Dokument.

Genau wie Sie in einem Word-Dokument nichts reinschreiben können, wo noch nichts ist, kann der Startbereich beim Einfügen von Text mit Word-VBA nicht grösser sein als der **Offset** des letzten Zeichen (oder um es genauer zu sagen nicht grösser als der Offset der letzten Absatzmarke). Möchten Sie an eine hypothetische (noch nicht vorhandene) Stelle weiter unten im Dokument gelangen, dann müssen Sie allenfalls wie bei einer manuellen Texteingabe zuerst Absatzmarken oder Zeilenumschaltungen einfügen.

Aber wie bereits erwähnt, es ist nicht die Idee, dass Sie nun die Zeichen im Dokument abzählen sollten. Wenn MS Word keine Strukturen von sich aus bereit stellt, dann schaffen Sie sich selbst solche Gefässe. Hier eigenen sich Textmarken oder auch konventionelle Formularfelder. Auch Textfelder oder Tabellen können dafür herhalten. All diese Elemente können Sie dann über deren Range-Eigenschaft adressieren. Das am nächsten liegende, flexibelste und einfachste Objekt bezüglich Handhabung in diesem Sinne, ist wohl eine Textmarke. Zum Thema Textmarken gibt es ein eigenes Skript in dieser Reihen. Beachten Sie dazu den entsprechenden Link, welcher am Ende dieser Abhandlung aufgeführt ist an.

Verzeichnis der hier präsentierten VBA-Lösungen bzw. Lösungsansätze

▲ Zurück zum Anfang

In der folgende Tabelle finden Sie eine Auflistung einiger Code-Beispiele, welche in dieses Skript diskutiert werden. Dabei wurden nicht alle im Skript vorhandenen VBA-Splitter aufgelistet sondern lediglich jene, welche den Charakter einer Lösung für eine bestimmte Aufgabenstellung aufweisen, oder zu mindest als Lösungsansatz dazu dienen können.

Dieses Skript enthält also noch zahlreiche weitere Code-Beispiele, meistens einfacher Natur, welche in der Regel genau auf eine Problematik (technische *Finesse*) eingehen und Sie Schritt für Schritt an die Sache heranführen.

Aufgabenstellung	Beschreibung
▼ Text zwischen zwei Textmarken markieren.	Mimik, um auf den Text zwischen zwei Textmarken zuzugreifen.
▼ Alle Kopf-/und Fusszeilen in sämtlichen Abschnitten löschen	In allen Abschnitten des Dokumentes werden die Kopf-/und Fusszeilen gelöscht. Dies umfasst primäre, erste Seite und gerade Seite Kopf-/und Fusszeile.
▼ Komplexe Fusszeile (inkl. Feldfunktionen) zusammenbauen	Eine Fusszeile wird mittels VBA-Code erstellt. Sie besteht aus drei Teilen. Linksbündig wird die Feldfunktion Author eingefügt, dann zentriert die Feldfunktion CreateDate (bzw. ErstellDat in Word 97) und rechtsbündig der Feldausdruck Seite x von y.
▼ Seitenzahlen in allen Kopf- und Fusszeilen löschen	In sämtlichen Kopf- und Fusszeilen werden für alle Abschnitte des Dokumentes, Seitenzahlen gelöscht, sollten diese dort auftreten.
▼ Faltmarke in das Dokument einbringen	Eine Faltmarke wird in das Dokument eingebracht. Damit dieses Gebilde (es handelt sich faktisch um eine Shape) nicht unbeabsichtigt vom Benutzer verschoben werden kann, sorgen wir dafür, dass die Marke in der Kopfzeile verankert wird. Sie erscheint allerdings trotzdem am richtigen Ort im Dokument. Wir richten es so ein, dass die Marke ausschliesslich auf der ersten Seite des Dokumentes erscheint.
▼ Kommentare in formatierten Text umwandeln	Statt am Ende des Dokumentes - beim Ausdruck - erscheinen Kommentare als formatierten Text und zwar an der Stelle, wo sie am Bildschirm eingegeben werden.
▼ Kommentare in Fussnoten umwandeln.	Die Kommentare im Dokument werden in Fussnoten umgewandelt.
▼ Tabulator vor jeden Fussnotentext setzen	Setze vor jeden Fussnotentext ein Tabulator.
▼ Inhaltsverzeichnis der Fussnoten erstellen	Am Schluss des Dokumentes werden nochmals alle Fussnoten aufgeführt, und zwar inklusive Seitenangabe und Referenznummer der Fussnote.
▼ Tabellenspalte durchsuchen	Für eine Tabellenspalte wird ein vorgegebenes Wort mit einem anderen Wort vertauscht
▼ Zählen, wie oft ein Wort im Dokument vorkommt	Der Code berücksichtigt beim Zählen alle Dokumententeile, also auch Kopf- und Fusszeilen, Textfelder, etc.
▼ Wörter für ein Benutzerwörterbuch automatisiert erfassen	Wörter in einer Sammlung von Dokumenten, welche der Rechtschreibprüfung von MS Word unbekannt sind, werden automatisiert erfasst und dann in ein Benutzerwörterbuch eingetragen
▼ Felder in allen Dokumententeilen aktualisieren	Mimik um Felder auch in den Kopf- und Fusszeilen, Textfelder etc. zu aktualisieren.
▼ Formatierten Text transferieren	Mimik um Text innerhalb oder zwischen Dokumenten zu transferieren unter Erhaltung der Formatierung und ohne Benutzung der Zwischenablage.
▼ Eine bestimmte Seite markieren	Mimik um auf den Text einer bestimmten Seite zuzugreifen.

Tabelle der Word-Objekte, welche über eine Range-Eigenschaft verfügen

▲ Zurück zum Anfang

Sie finden anschliessend eine Tabelle, welche die wichtigsten Word-Objekte, die über eine *Range*-Eigenschaft verfügen, auflistet. Die Tabelle (und somit auch dieses Skript) ist nicht ganz, aber annähernd vollständig. Wenn Sie sich für eines der Objekte näher interessieren, gelangen Sie mit einem Klick auf den entsprechenden Link direkt zu den Beispielen, welche den Einsatz der *Range*-Eigenschaft für dieses Objekt genauer erläutern.

Object	Objekt	Beispiel	Bemerkungen
▼ Selection	Markierung	`Set oRange = Selection.Range`	Das *Selection*-Objekt stellt die aktuelle Markierung dar. Ist nichts markiert, wird die Position der Einfügemarke zurück gegeben.

▼ Bookmark	Textmarke	`If oDoc.Bookmarks.Exists("MeineMarke") Then`
		`  oDoc.Bookmarks("MeineMarke").Range.Text = "Hallo"`
		`End If`
			Bei einer offenen Textmarke wird eine Position angesprochen, wogegen bei einem Textmarkenfeld ein Bereich angesprochen wird.

▼ Doucument	Dokument	`oDoc.Range.InsertAfter "Hallo Welt"` `oDoc.Content.InsertAfter "Hallo Welt"`	Die Begriffe *oDoc.Range* und *oDoc.Content* scheinen synonym zu sein.
▼ Section	Abschnitt	`nDoc.Content.FormattedText = _` `  oDoc.Sections(1).Range.FormattedText`	Im Gegensatz zu der Text-Eigenschaft übernimmt die *FormattedText*-Eigenschaft auch die Formatierung.
▼ Paragraph	Absatz	`strText = oDoc.Paragraphs(1).Range.Text`	Sie können auch einen Bereich aus sich folgenden Absätzen bilden.

▼ Header, Footer	Kopfzeile, Fusszeile	`Set oRange = _`
		`oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range`
		`oRange.Text = Application.UserName`
			Jeder Abschnitt eines Dokumentes kann bis zu drei Kopf-/Fusszeilenbereiche aufweisen (*Haupt-*, *Erste Seite-*, *Gerade* Seite-Kopf/Fusszeile)

▼ TextFrame	Textfeld	`MsgBox oDoc.Shapes(1).TextFrame.TextRange.Text`	Beachten Sie, dass ein Textfeld indirekt über das entsprechende *Shape*-Objekkt angesprochen wird. Beachten Sie ebenfalls, dass es *TextRange* an Stelle von *Range* lauten muss.
▼ InlineShape	Grafik, Objekt (textgebunden)	`oDoc.InlineShapes(1).Range.Font.Hidden = True`	Mit diesem Code kann eine Grafik oder ein Objekt, welches als *InlineShape* eingebracht wurde, versteckt werden.
▼ Comment	Kommentar	`MsgBox oDoc.Comments(1).Range.Text` `Set oRange = oDoc.Comments(1).Scope`	Das erste Beispiel gibt den Kommentar als solches zurück. Dieser befindet sich in einer anderen Ebene (*StoryRange*). Das zweite Beispiel gibt den Bereich, in dem der Kommentar eingefügt wurde, zurück. Die *Scope*-Eigenschaft gibt also auch ein *Range*-Objekt zurück.

Object	Objekt	Beispiel	Bemerkungen
▼ Footnote	Fussnote	`MsgBox oDoc.Footnotes(1).Range.Text` `Set oRange = oDoc.Footnotes(1).Reference`	Das erste Beispiel gibt die Fussnote als solches zurück. Dieser befindet sich auf einer anderen Ebene (*StoryRange*). Das zweite Beispiel gibt den Bereich, in dem der Fussnote eingefügt wurde, zurück. Die *Reference*-Eigenschaft gibt also auch ein *Range*-Objekt zurück.
▼ Endnote	Endnote	`MsgBox oDoc.Endnotes(1).Range.Text` `Set oRange = oDoc.Endnotes(1).Reference`	siehe Fussnote
▼ Field	Feld	`MsgBox oDoc.Fields(1).Code.TextMsgBox` `MsgBox oDoc.Fields(1).Result.Text`	Benutzen Sie die Eigenschaft *Code* (statt *Range*) um den Feldcode zurückzugeben. Benutzen Sie die Eigenschaft *Result* (statt *Range*) um das Ergebnis der Feldfunktion zurückzugeben.
▼ FormField	Formularfeld	`Set oRange = oDoc.FormFields("Text1").Range` `MsgBox oDoc.FormFields("Text1").Result`	Verwenden Sie die *Result*-Eigenschaft (statt *Range*), um den Wert eines Formulartextfeldes zurück zu geben.

▼ Table	Tabelle	`i = 0`
		`Dim Zelle as Cell`
		`For Each Zelle In oDoc.Tables(1).Range.Cells`
		`  i = i + 1`
		`  Zelle.Range.Text = i`
		`Next`
			--- 	

▼ Cell	Tabellenzelle	`oDoc.Tables(1).Range.Cells(1).Range.Font.Bold = True`	---
▼ Find	Suchen/ Ersetzen	`With oDoc.Range.Find` `End With`	Als einzige Ausnahme soll *Selection.Find* *Range.Find* bezüglich Tempo überlegen sein.
▼ Frame	Positionsrahmen	`MsgBox oDoc.Frames(1).Range.Text`	---
▼ SpellingError	Rechtschreibfehler	`MsgBox oDoc.SpellingErrors(1).Text`	Das *SpellingError*-Objekt gibt also einen *Range* zurück.
▼ StoryRange	Dokument-Ebene	`nDoc.Content.FormattedText =` `  oDoc.StoryRanges(wdEndnotesStory).FormattedText`	Das Word-Objekt-Modell definiert elf verschiedene *StoryRanges*, welche in einer Tabelle am Ende dieses Skriptes zusammengefasst sind.
▼ Line (kein richtiges Objekt)	Zeile	`MsgBox oDoc.Bookmarks("\Line").Range.Text`	Beachten Sie, dass eine Zeile kein Objekt im Sinne des Word-Objekt-Models darstellt. Deshalb können Sie Zeilen nur über Word-integrierten Textmarken ansprechen.

▼ Page (kein richtiges Objekt)	Seite	`Selection.GoTo What:=wdGoToPage, _`
		`  which:=wdGoToAbsolute, Count:=3 Selecti-`
		`on.Bookmarks("\Page").Select`
			Genau wie Zeilen können auch Seiten nur über die word-imtegrierten Textmarken angesprochen werden.

Die Tabelle zeigt einerseits auf, dass sehr viele Word-Objekte über die *Range*-Eigenschaft verfügen. Allerdings fällt auch auf, dass zwei Elemente, die in der Textverarbeitung eine Bedeutung haben fehlen, nämlich Zeilen und Seiten.

Dieses Fehlen ist kein Zufall, hat es doch mit dem Word-Konzept zu tun, in welchem Zeilen und Seiten keine Objekte darstellen und andererseits mit Fliesstext, wo Zeilen- und Seitenumbrüche fliessend sind. Lesen Sie dazu das Arbeiten mit Zeilen und Seiten in diesem Skript.

Ranges ansprechen

▲ Zurück zum Anfang

Nach der oben aufgeführten Tabelle der Objekte, welche über die *Range*-Eigenschaft verfügen, folgen nun für jedes Objekt typische Beispiele. Dabei wird aufgezeigt, wie Sie die *Range*-Eigenschaft der einzelnen Objekte mittels Word-VBA ansprechen bzw. einsetzen können können.

Das Selection-Objekt

▲ Zurück zur Übersicht der Objekte

Das *Selection*-Objekt umfasst den Bereich, welcher gerade markiert ist. Ist nichts markiert, dann erhält man so die Position der Einfügemarke zurück, sonst den markierten Bereich.

Um eine fehlerfreie Abwicklung des Codes zu garantieren, sollten Sie zuerst feststellen, was markiert ist. Diese Information erhalten Sie mit der ***Type***-Eigenschaft des *Selection*-Objekts zurück.

```
Sub MerkierterTextZuruechgeben()
  If Selection.Type = wdSelectionNormal Then
    MsgBox Selection.Range.Text
  Else
    MsgBox "Es ist kein Text markiert."
  End If
End Sub
```

(Der markierte Text wird zurückgegeben.)

Der Ausdruck **`Selection.Text`** würde übrigens das selbe Resultat wie **`Selection.Range.Text`** abliefern, wäre aber vom Prinzip her nicht ganz so sauber. Wie auch immer, Sie können diese beiden Ausdrücke synonym verwenden. Diese Doppelspurigkeit hat damit zu tun, dass VBA für die meisten Objekte eine Standardeigenschaft kennt. Sehr oft ist der *Range* diese Standardeigenschaft.

Wie schon erwähnt, Sie sollten testen um welche Art von Markierung es sich handelt. Im Folgenden finden Sie eine Tabelle, welche die Werte aufzeigt, mit denen Sie dabei rechnen müssen.

Konstante	Wert	Bedeutung
WdSelectionNormal	2	Die Einfügemarke befindet sich im Text und es ist Text markiert (mindestens ein Zeichen)
WdSelectionIP	1	Die Einfügemarke befindet sich im Text. Es ist allerdings kein Text markiert.
WdSelectionFrame	3	Der Rand eines Positionsrahmens ist markiert.
WdSelectionShape	8	Eine ungebundene (schwebende) Grafik ist markiert. Dabei kann es sich auch um ein Textfeld handeln.
wdSelectionInlineShape	7	Eine textgebundene Grafik ist markiert
WdSelectionBlock	6	Wenn der Benutzer die **[ALT]** Taste gedrückt hält und den Mauszeiger über den Text zieht, kann er einen Block markieren.
WdSelectionRow	5	Eine Zeile einer Word-Tabelle ist markiert.
WdSelectionColumn	4	Eine Spalte einer Word-Tabelle ist markiert.
WdNoSelection	0	Es ist nicht markiert und die Einfügemarke befindet sich auch nicht im Text.

Im folgendem Beispiel, wird ein Bereich (*Range*) gebildet, welcher die Markierung darstellt. Angenommen, Sie heben die Markierung in Ihrem Code auf, wollen dann aber später wieder darauf zurückkommen, also die ursprüngliche Markierung wieder herstellen, dann können Sie sich auf das erstellte *Range*-Objekt beziehen und folgendes kodieren: **`oRange.Select`**.

```
Sub SelectionNachRange()
  Dim oRange As Range
  Set oRange = Selection.Range
  'Ihr Code
  'Ihr Code
  'Ihr Code
  oRange.Select
End Sub
```

(Es wird ein Range gebildet, welcher die Markierung darstellt.)

Das Bookmark-Objekt

▲ Zurück zur Übersicht der Objekte

Beim Objekt Textmarke in MS Word ist zu beachten, dass es eigentlich zwei Arten von Textmarken bzw. reine Textmarken und Textmarkenfelder gibt. Die erste Art markiert eine Position im Dokument und hat selbst keinen Inhalt (**offene Textmarke**). Die zweite Art markiert einen Bereich im Dokument und verfügt über einen Inhalt bzw. Länge. Diese Textmarken erscheinen als geschlossene Textmarken und werden offiziell **Textmarkenfelder** genannt

Entsprechend dieser Unterscheidung gibt die Abfrage des *Ranges* einer offenen Textmarke eine Position (Anfang und Ende sind gleich) und das Textmarkenfeld einen wirklichen Bereich zurück. Bei einer offenen Textmarke, erhalten Sie also auch kein Text zurück.

Wegen der hohen Flexibilität, welche Textmarken in der Word-VBA-Programmierung bieten, wird dieses Thema in einem zusätzlichen Skript dieser Reihe abgehandelt. Den Link dazu finden Sie am Ende dieses Kapitels.

Im Folgenden wird bei der Textmarke mit dem Namen **Ort** Text beigefügt:

```
Sub TextBeiTextmarkeEinbringen1()
  ActiveDocument.Bookmarks("Ort").Range.Text = "Basel"
End Sub
```

(Bei der Textmarke mit dem Namen Ort wird Text eingefügt.)

Es ist eine gute Praxis zuerst zu prüfen, ob eine Textmarke, welche angesprochen werden soll auch wirklich existiert. Bei Missachtung und unter der Annahme, dass die Textmarke nicht vorhanden ist, bricht der Code nämlich unkontrolliert ab. Deshalb das selbe Beispiel nochmals, diesmal aber mit Prüfung.

```
Sub TextBeiTextmarkeEinbringen2()
  BMName = "Ort"
  If ActiveDocument.Bookmarks.Exists(BMName) Then
    ActiveDocument.Bookmarks(BMName).Range.Text = "Basel"
  End If
End Sub
```

(Prüfen statt Russisches Roulette.)

Das nächste Beispiel liest den Text, des Textmarkenfeldes mit dem Namen **Ort** aus. Sinnigerweise sollte es sich dabei um ein Textmarkenfeld handeln, denn eine offene Textmarke hat keinen Inhalt, sonder markiert einen Punkt im Dokument.

```
Sub TextmarkenfeldAuslesen()
  MsgBox ActiveDocument.Bookmarks("Ort").Range.Text
End Sub
```

(Liesst das Textmarkenfeld mit dem Namen Ort aus.)

Text zwischen zwei Textmarken markieren

▲ Zurück zu den Aufgabenstellungen

Im folgenden wird der Text markiert, welcher bei der Textmarke mit dem Namen **MeinAnfang** beginnt und mit der Textmarke **MeinEnde** endet. Meiner Meinung nach, brauchen Sie diese Mimik in der Praxis eigentlich gar nicht. Wenn Sie mit einem Textmarkenfeld statt zwei offenen Textmarken operieren, dann liegt Ihnen bereits ein Bereich vor, Sie brauchen nur eine Textmarke und die Syntax wird bedeutend einfacher.

```
Sub TextZwischenZweiTextmarkenMarkieren()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  If oDoc.Bookmarks.Exists("MeinAnfang") And _
     oDoc.Bookmarks.Exists("MeinEnde") Then
    oDoc.Range(Start:=oDoc.Bookmarks("MeinAnfang").Range.Start, _
       End:=oDoc.Bookmarks("MeinEnde").Range.Start).Select
  End If
End Sub
```

(Markiert eine Textstelle zwischen zwei offenen Textmarken.)

Zusätzliche Informationen zum Arbeiten mit Textmarken erhalten Sie im folgenden Skript:

► VBA-technischer Umgang mit Textmarken

Das Document-Objekt

▲ Zurück zur Übersicht der Objekte

Der Range des *Document*-Objektes erstreckt sich über den Hauptteil des Dokumentes. Nicht eingeschlossen sind alle übrigen Dokumentteile (Ebenen), in der Fachsprache *StoryRanges* genannt. Schauen Sie sich dazu das Kapitel Tabelle der Dokumentteile (*StoryRanges*) in MS Word gegen Ende dieses Skriptes an.

Es scheint, dass die beiden Begriffe **`ActiveDocument.Range`** und **`ActiveDocument.Content`** synonym behandelt werden können. Die Eigenschaft *Content* gibt hier also auch ein *Range*-Objekt zurück.

Sie können den Bereich aber auch eingrenzen, indem Sie einen Start- und End-Bereich angeben. Damit wird es dann wirklich interessant und das Können bzw. die Kunst beginnt. Beispiele dazu finden Sie etwas weiter unten im Skript. Im folgenden Beispiel wird Text am Ende des aktiven Dokumentes eingefügt.

```
Sub TextAmEndeDesHautteilsDesDokumentesEinfuegen()
  ActiveDocument.Range.InsertAfter "Hallo Welt"
End Sub
```

(Am Ende des Hauptteils des Dokumentes wird Text eingefügt.)

Im folgenden Beispiel wir der Text im Hauptteil des Dokuments in einen neues Dokument kopiert. Beachten Sie, dass dabei die Formatierung des Textes nicht mit wandert. Wie man Text mit Formatierung transferieren kann, erfahren Sie im Kapitel Arbeiten mit *FormattedText* weiter unten in diesem Skript.

```
Sub TextDesHauptteilsInNeuesDokumentKopieren()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add
  nDoc.Content.Text = oDoc.Content.Text
End Sub
```

(Der Text im Hauptteil des aktiven Dokumentes wird in ein neues Dokument kopiert..)

Im folgenden wird der Anfang des Hauptteiles des Dokumentes markiert. Da die Länge des angegebenen Bereich 0 ist, wird aber faktisch nichts markiert, vielmehr wird die Einfügemarke verschoben. Somit stelle das Beispiel eine Alternative zum Ausdruck **`Selection.HomeKey Unit:=wdStory`** dar.

```
Sub EinfügemarkeAnDenAnfangDesDokumentes1()
  ActiveDocument.Range(Start:=0, End:=0).Select
End Sub
```

(Die Einfügemarke wird unter Umgehung des Selection-Objektes an den Anfang des Hauptteiles des Dokumentes gesetzt.)

Das Gleiche ist ohne Schaden auch mit einem etwas kürzeren Code zu erreichen.

```
Sub EinfügemarkeAnDenAnfangDesDokumentes2()
  ActiveDocument.Range(0, 0).Select
End Sub
```

(Wie des vorherige Beispiel, allerdings etwas kürzer, was die Syntax betrifft.)

Im folgenden Beispiel werden die **Zeichen 3-5** im Hauptteil des Dokumentes markiert. Beachten Sie, dass die Zählung bei 0 und nicht bei 1 beginnt. Das Beispiel hat wahrscheinlich keine allzu grosse praktische Bedeutung, da Sie sich besser an markanteren Objekten wie z.B. Absätze oder Tabellen orientieren sollten, zeigt aber nochmals auf, was ein *Range* darstellt, bzw. wie dieser eingesetzt werden kann. Sollten Orientierungshilfen rar sein, schaffen Sie solche, in dem Sie Textmarken in das Dokument einbringen. Beachten Sie, dass nicht nur Text zählt, sondern auch Steuerzeichen, wie z.B. die Absatzmarke.

```
Sub BereichMarkieren()
  ActiveDocument.Range(Start:=2, End:=4).Select
End Sub
```

(Eine Zusammenhängende Zeichenfolge im Hauptteil des Dokumentes wird markiert.)

! Mehrfachmarkierung von Text, also das Markieren von mehreren nicht zusammenhängende Textstellen, ist in MS Word übrigens nicht möglich, auch nicht unter Einsatz von VBA. Als Ausnahmen dieser Regel kann das Markieren von Tabellenspalten und die Möglichkeit einen Block zu markieren (bei gedrückter **[Alt]**-Taste) gesehen werden.

Das Section-Objekt

▲ Zurück zur Übersicht der Objekte

Jedes Dokument verfügt über mindestens einen Abschnitt. Zusätzliche Abschnitte entstehen, wenn Sie den Befehl *Einfügen - Manueller Wechsel - Abschnittwechsel* einsetzten.

Vor allem Eigenschaften, welche im Zusammenhang mit dem Seitenlayout stehen und Kopf- und Fusszeilen sind vom *Section*-Objekt abhängig. Daneben verfügt das *Section*-Objekt selbstverständlich auch über die *Range*-Eigenschaft.

Im folgenden Beispiel wird der Text des ersten Abschnittes des Dokumentes in ein neues Dokument kopiert. Die Formatierung bleibt dabei erhalten. Sollte das Dokument nur über einen Abschnitt verfügen, wird faktisch der ganze Hauptteil des Dokumentes kopiert.

```
Sub AbschnittMitFormatierungKopieren()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add(Template:=oDoc.AttachedTemplate.FullName)
  nDoc.Content.FormattedText = oDoc.Sections(1).Range.FormattedText
End Sub
```

(Der erste Abschnitt des aktuellen Dokumentes wird - mit Formatierung - in ein neu erstelltes Dokument kopiert.)

Sämtliche Kopf- und Fusszeilenbereiche in einem Dokument löschen

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiele werden sämtliche Kopf- und Fusszeilenbereiche des Dokuments gelöscht unter Berücksichtigung der Tatsache, dass das Dokument über mehrere Abschnitte verfügen kann.

```
Sub AlleKopfundFusszeilenLöschen()
  Dim Abschnitt As Section
  For Each Abschnitt In ActiveDocument.Sections
    For i = 1 To 3
      Abschnitt.Headers(i).Range.Delete
      Abschnitt.Footers(i).Range.Delete
    Next i
  Next
End Sub
```

(Löscht sämtliche Kopf- und Fusszeilen im Dokument. Dies geschieht für alle Abschnitte.)

Das Paragraph-Objekt

▲ Zurück zur Übersicht der Objekte

Im Folgenden finden Sie Beispiele, wie die *Range*-Eigenschaft eines Absatzes (englisch *paragraph*) angesprochen werden kann. Wegen des hohen Stellenwertes diese Objektes erfahren Sie in einem zusätzlichen Skript dieser Reihe mehr über dessen Einsatz. Sie finden den Link zu diesem Skript am Ende dieses Kapitels Das zusätzliche Skript enthält auch Informationen und Beispiele, wie Absätze in einem Dokument sequentiell (also von oben bis unten) abgearbeitet werden können.

Im folgenden wird der erste Absatz (Absatzmarke inklusive) des Dokumentes in eine Variable geladen:

```
Sub ErsterAbsatzInVariable()
  strText = ActiveDocument.Paragraphs.First.Range.Text
End Sub
```

(Der erste Absatz im Dokument wird in eine Variable geladen.)

Im folgenden Beispiel wird ein *Range* gebildet, welche den dritten Absatz (exklusive Absatzmarke) beinhaltet. Beachten Sie, dass standardmässig die Absatzmarke mit inbegriffen ist. Wollen Sie dies vermeiden, dann müssen Sie am Ende des Bereiches das Steuerzeichen abziehen, wie das im Beispiel vorexerziert wird. Sollte das Dokument über weniger als drei Absätze verfügen, bricht der Code mit einer Fehlermeldung ab.

```
Sub MehrereAbsaetzeNachRangeOhneAbsatzMarke()
  Dim oDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  Set oRange = oDoc.Range(Start:=oDoc.Paragraphs(3).Range.Start, _
     End:=oDoc.Paragraphs(3).Range.End - 1)
  oRange.Select
End Sub
```

(Bildung eines Ranges, welcher den dritten Absatz ohne Absatzmarke umfasst.)

Sie können auch mehrerer Absätze, welche direkt aufeinander folgen ansprechen. Im nächsten Beispiel wird ein Range gebildet, welcher die **Absätze 3-6** des Dokumentes umfasst:

```
Sub MehrereAbsätzeNachRange()
  Dim oDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  Set oRange = oDoc.Range(Start:=oDoc.Paragraphs(3).Range.Start, _
     End:=oDoc.Paragraphs(6).Range.End)
  oRange.Select
End Sub
```

(Aus den Absätzen 3-6 des Dokumentes wird ein Range gebildet.)

Zusätzliche Informationen zum Arbeiten mit Absätzen, einschliesslich die sequentielle Verarbeitung von Absätzen, erhalten Sie im folgenden Skript:

► VBA-technischer Umgang mit Absätzen

Das Header- und das Footer-Objekt

▲ Zurück zur Übersicht der Objekte

Beachten Sie, dass Kopf- und Fusszeilen auf einer anderen Dokumentenebene (*StoryRange*) als der Hauptteil des Dokumentes liegen. Beachten Sie auch, dass das *Parent*-Objekt von Kopf- oder Fusszeilen nicht das Dokument selbst ist, sondern die einzelnen Abschnitte (*Sections*) eines Dokumentes. Sie können in einem Dokument ja potentiell für jeden Abschnitt eine andere Kopf-/Fusszeile definieren. Voraussetzung dazu, ist dass die Eigenschaft Wie vorherige ausgeschaltet ist.

Es kommt aber noch bunter. Pro Abschnitt sind je drei Arten von Kopf- bzw. Fusszeilen möglich, nämlich die **Primäre**, die Kopf-/Fusszeile für die **Erste Seite** und schliesslich die Kopf-/Fusszeile für **Gerade Seiten**. Dabei handelt es sich um ein Potential. Diese zusätzlichen Dokumententeile stehen ausschliesslich dann zur Verfügung, wenn die Option *Erste Seite anders* bzw. *Gerade-/Ungerade Seiten* eingeschaltet ist.

Von den elf *StoryTypes*, welche das Word-Objekt-Model definiert, stehen also nicht weniger als sechs in Verbindung mit Kopf-/und Fusszeilen. Eine komplette Auflistung aller *StoryTypes* finden Sie in einer Tabelle am Ende dieses Kapitels.

Ausgerüstet mit diesen Informationen, können wir nun erahnen, wie die Syntax aussehen muss, um einen Kopf- oder Fusszeilenbereich anzusprechen. Dazu ist es übrigens nicht nötig, die Ansicht im Dokument zu wechseln, also in die Kopf-/Fusszeilenansicht zu schalten.

Im folgenden Beispiel wird Text in die primäre Kopfzeile des ersten Abschnittes des Dokumentes eingefügt. Allfälliger Text, welcher dort schon vorhanden ist, wird dabei überschrieben.

```
Sub TextInPrimaereKopfzeileDesErstenAbschnittesEinfuegen1()
  ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Text = _
     "Hallo Welt"
End Sub
```

(Fügt Text in die primäre Kopfzeile des ersten Abschnittes ein.)

Nachstehend funktionell das selbe Beispiel wie oben, allerdings etwas handlicher codiert.

```
Sub TextInPrimaereKopfzeileDesErstenAbschnittesEinfuegen2()
  Dim Kopf As HeaderFooter
  Set Kopf = ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary)
  Kopf.Range.Text = "Hallo Welt"
End Sub
```

(Eine Variante zum obigen Beispiel.)

Im nachfolgenden Beispiel wird eine Feldfunktion in die primäre Fusszeile des Dokumentes eingesetzt. Anschliessen werden die Felder in der Fusszeile aktualisiert

```
Sub FeldInPrimaereFusszeileDesErstenAbschnittesEinfuegen2()
  Dim Fuss As HeaderFooter
  Set Fuss = ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary)
  ActiveDocument.Fields.Add Range:=Fuss.Range, _
     Type:=wdFieldFileName, Text:="\p"
  Fuss.Range.Fields.Update
End Sub
```

(Eine Feldfunktion wird in die primäre Fusszeile des ersten Abschnittest im Dokuments eingefügt.)

Im nächsten Beispiel wird zuerst dafür gesorgt, dass der erste Abschnitt im Dokument zusätzlich den Bereich Fusszeile Erste Seite erhält. Anschliessend wird Text in diesen Bereich geschrieben.

```
Sub FusszeilenBereichAnsprechen()
  Dim oDoc As Document, Fuss As HeaderFooter
  Set oDoc = ActiveDocument
  oDoc.Sections(1).PageSetup.DifferentFirstPageHeaderFooter = True
  Set Fuss = oDoc.Sections(1).Footers(wdHeaderFooterFirstPage)
  Fuss.Range.Text = "Das ist die Fusszeile der ersten Seite."
End Sub
```

(Die Erste Seite Fusszeile wird gebildet und dann mit Text bestückt.)

Komplexe Fusszeile erstellen

▲ Zurück zu den Aufgabenstellungen

Im nächsten Beispiel, wird eine Fusszeile mittels VBA-Code erstellt, welche aus drei Teilen besteht. Dabei wird **linksbündig** die Feldfunktion **Author** eingefügt, dann - **mittig** - die Feldfunktion ***CreateDate*** (bzw. *ErstellDat* in Word 97) und rechtsbündig der Feldausdruck **Seite x von y**.

Diese Arbeiten geschehen in der primären Kopfzeile des ersten Abschnittes des Dokumentes. Allfällige bestehende Inhalte der Fusszeile werden dabei überschrieben.

Die Routine setzt auch die *SetRange*-Methode ein. Diese dient dazu Start- oder End- Punkt eines Ranges zu ändern. Die SetRange-Methode wird in diesem Skript noch genauer besprochen.

```
Sub FusszeileZusammenbauen()
  If Application.Version = "8.0" Then 'Word 97
    strDatum = "\@" & Chr(34) & "t. MMMM jjjj" & Chr(34)
  Else 'Word2000 und höher
    strDatum = "\@" & Chr(34) & "d. MMMM yyyy" & Chr(34)
  End If
  Dim oDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.Delete 'Fusszeilenbereich löschen
  With oDoc.Sections(1).PageSetup 'Breite des bedruckbaren Bereiches
    Breite = .PageWidth - .LeftMargin - .RightMargin
  End With
  With oRange.ParagraphFormat
    .TabStops.ClearAll 'Tabstopps löschen
    .TabStops.Add Position:=0 'Tabstopp links
    .TabStops.Add Breite / 2, Alignment:=wdAlignTabCenter 'Tabstopp Mitte
    .TabStops.Add Position:=Breite, Alignment:=wdAlignTabRight 'Tabstopp rechts
  End With
  oRange.Fields.Add Range:=oRange, Type:=wdFieldAuthor 'Author
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.InsertAfter vbTab
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  oRange.Fields.Add Range:=oRange, Type:=wdFieldCreateDate, Text:=strDatum
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.InsertAfter vbTab & "Seite "
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  oRange.Fields.Add Range:=oRange, Type:=wdFieldPage
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.InsertAfter " von "
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  oRange.Fields.Add Range:=oRange, Type:=wdFieldNumPages
End Sub
```

(Komplexer Fusszeilentext wird in den Fusszeilenbereich eingefügt, ohne in die entsprechende Ansicht zu wechseln.)

Sämtliche Seitenzahlen löschen

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiel werden in sämtlichen Kopf- und Fusszeilen für alle Abschnitte des Dokumentes, die Seitenzahlen gelöscht, sollten diese dort auftreten.

```
Sub AlleSeitenzahlenInKopfUndFusszeileLöschen()
  Dim Abschnitt As Section
  Dim Feld as Field
  For Each Abschnitt In ActiveDocument.Sections
    For i = 1 To 3
      For Each Feld In Abschnitt.Headers(i).Range.Fields
        If Feld.Type = 26 Or Feld.Type = 33 Then Feld.Delete
      Next
      For Each Feld In Abschnitt.Footers(i).Range.Fields
        If Feld.Type = 26 Or Feld.Type = 33 Then Feld.Delete
      Next
    Next i
  Next
End Sub
```

(Löscht Seitenzahlen in sämtlichen Kopf- und Fusszeilen des Dokumentes.)

Faltmarke einbringen

▲ Zurück zu den Aufgabenstellungen

Im letzten Beispiel dieses Kapitels wird eine Faltmarke in das Dokument eingebracht. Damit dieses Gebilde (es handelt sich faktisch um eine *Shape*) nicht unbeabsichtigt vom Benutzer verschoben wird, sorgen wir dafür, dass die Marke in der Kopfzeile verankert wird. Sie erscheint allerdings trotzdem am richtigen Ort im Dokument. Wir richten es ausserdem so ein, dass die Marke ausschliesslich auf der ersten Seite des Dokumentes erscheint. Dies ist somit ein Fall für die *Erste Seite Kopfzeile*.

```
Sub FaltmarkeEinfügenNurErsteSeite()
  'Die Länge des Striches beträgt 0,9 - 0,5 = 0,4 cm
  Dim oKz As HeaderFooter, FM As Shape
  Rand = ActiveDocument.Sections(1).PageSetup.LeftMargin
  ActiveDocument.Sections(1).PageSetup.DifferentFirstPageHeaderFooter = True
  Set oKz = ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage)
  Set FM = oKz.Shapes.AddLine(CentimetersToPoints(0.5) - Rand, _
                              CentimetersToPoints(10.4), _
                              CentimetersToPoints(0.9) - Rand, _
                              CentimetersToPoints(10.4), oKz.Range)

  FM.Line.Weight = 0.25
  FM.LockAnchor = True
End Sub
```

(Fügt eine Faltmarke in das Dokument ein. Dieser Hilfsstriche erscheint ausschliesslich auf der ersten Seite des Dokumentes.)

Das TextFrame-Objekt (Textfeld)

▲ Zurück zur Übersicht der Objekte

Ein *TextFrame*-Objekt entsteht dann, wenn Sie z.B. über die Symbolleiste mit der Bezeichnung *Zeichnen* ein **Textfeld** in das Dokument einbringen.

Bei Textfeldern ist zu berücksichtigen, dass diese in erster Line sogenannte ***Shapes*** darstellen. Das Textfeld ist dann eines der möglichen Unterobjekte einer Shape. Das Textfeld seinerseits hat dann tatsächlich eine *Range*-Eigenschaft, welche allerdings als ***TextRange*** angesprochen wird.

Ausser was die Syntax betrifft, braucht es Sie nicht gross zu kümmern, dass die Eigenschaft *TextRange* heisst. Was nämlich zurückkommt ist ein veritabeler *Range*, welchen Sie auch als solchen ansehen und behandeln können.

Schon mehr Kopfzerbrechen bereitet hier die Frage, wie finde ich genau jenes Textfeld, welches ich ansprechen möchte. Wie wir bereits wissen, ist ein Textfeld in erster Linie eine *Shape*. Eine *Shape* hat einen Index, über welchen die *Shape* ansprechbar ist. Allerdings ist nicht jede *Shape* auch ein Textfeld, ergo ist die erste *Shape* im Dokument nicht zwingend das erste Textfeld im Dokument. Setzt man diese Erkenntnisse um, so findet man das erste Textfeld indem man die Sammlung der *Shapes* schleift und den Typ abfragt bis man auf das erste Textfeld stösst. Das folgende Beispiel demonstriert diese Mimik.

```
Sub ErstesTextfeldImHauptteilDesDokumentesErmitteln()
  Dim Form As Shape
  For Each Form In ActiveDocument.Shapes
    If Form.Type = msoTextBox Then
      Form.TextFrame.TextRange = "Hallo Welt"
    End If
  Next
End Sub
```

(Findet das erste Textfeld im Hauptteil des Dokumentes auf und schreibt Text in dieses Textfeld.)

Sie können ein Textfeld auch beim Namen ansprechen, welches das *Parent*-Objekt, also die *Shape* trägt. Beachten Sie dabei aber, dass diese Namen intern von Word vergeben werden und das die Namen etwas Zufälliges haben. Im folgenden Beispiel wird ein Textfeld angesprochen, dessen *Parent-Objekt Shape* die Bezeichnung **Text Box 2** trägt.

```
Sub TextFeldBeimNamenDerShapeAnsprechen()
  ActiveDocument.Shapes("Text Box 2").TextFrame.TextRange = "Mein Text"
End Sub
```

(Ein Textfeld wird über den Namen, welcher sein Parent-Objekt trägt, angesprochen.)

Sie können nun hingehen und der *Shape*, welche gleichzeitig das Textfeld darstellt, einen aussagekräftigen und eindeutigen Namen ihrer Wahl geben. Das erreichen Sie allerdings nur mittels VBA. Mit folgendem VBA-Code können Sie einem markierten Textfeld einen Namen vergeben. Achten Sie darauf, dass der **Rahmen** des Textfeldes und nicht dessen Inhalt markiert ist. Der Code überprüft zuerst, ob überhaupt eine *Shape* markiert ist. Dann prüft die Routine, ob es sich bei der markierten *Shape* auch wirklich um ein Textfeld handelt. Sind diese Bedingungen erfüllt wird der Name des Textfeldes geladen und in einer Input-Box zur Disposition gestellt. Abgefangen wird auch der Fall, welcher Eintritt, wenn der Benutzer einen Namen vergeben will, welcher bereits vergeben ist

```
Sub EinemTextfeldEinenNamenVergeben()
  If Not Selection.Type = wdSelectionShape Then
    MsgBox "Es ist kein Textfeld markiert"
    Exit Sub
  End If
  If Not Selection.ShapeRange(1).Type = msoTextBox Then
    MsgBox "es ist kein Textfeld markiert"
    Exit Sub
  End If
  Ist = Selection.ShapeRange(1).Name
  Antwort = InputBox("Geben Sie einen Namen für das Textfeld ein:", , Ist)
  If Antwort = "" Then Exit Sub
  On Error Resume Next
  Selection.ShapeRange(1).Name = Antwort
  If Err.Number = 70 Then
    MsgBox "Dieser Name ist bereits vergeben.", vbExclamation
    Exit Sub
  End If
  On Error GoTo 0
End Sub
```

(Erlaubt es dem Benutzer einem markierten Textfeld einen Namen zu vergeben.)

Wesentlich einfacher haben Sie es, wenn Sie mit Textmarken arbeiten. Um z.B. ein Textfeld mittels VBA mit Text zu bestücken, können Sie eine Textmarke in dieses Textfeld einbringen. Die Syntax, wie Sie nun den Text bei der Textmarke einfügen, ist unabhängig von der Tatsache, dass sich die Textmarke in einem Textfeld befindet. Beispiele, wie sie den Bereich einer Textmarke ansprechen finden Sie im Kapitel Das Bookmark-Objekt weiter oben in diesem Skript.

Das InlineShape-Objekt

▲ Zurück zur Übersicht der Objekte

Bei einer *InlineShape* handelt es sich um eine Objekt (oft eine Grafik), welche textgebunden in das Dokument eingebracht wurde. MS Word unterscheidet zwischen *Shapes* und *InlineShapes*. *Shapes* besitzen keine **Range**-Eigenschaft, denn sie sind nicht textgebunden sondern *floaten* (d. h. schweben) im Dokument.

Eine *InlineShape* entsteht in **Word 97** dann, wenn der Benutzer eine Grafik (oder ein anders Objekt einfügt) und dabei die Option *Über den Text legen deaktiviert*. In **Word2000** entsteht ein solches Objekt dann, wenn der Benutzer die Umbruchart *Mit Text in Zeile* wählt.

Wenn Sie mehr über *InlineShapes* bzw. den Unterschied zwischen *Shapes* und *InlineShapes* erfahren möchten, dann rate ich, dass Sie einen Blick in das entsprechende Skript aus dieser Reihe werfen. Den Link dazu finden Sie am Ende dieses Kapitels.

Das Auffinden einer bestimmten *InlineShape* im Dokument ist nicht immer einfach. Sie können eine *InlineShape* aber in jedem Falle über ihren Index ansprechen.

Eine weitere gute Möglichkeit, um eine InlineShape identifizierbar zu machen, sieht vor, dass Sie die *InlineShape* markieren und dann dieser Markierung eine Textmarke zuweisen. Über den Textmarkenname, welcher eindeutig ist, kann diese *InlineShape* später dann sehr leicht aufgefunden werden. Es handelt sich in diesem Fall um jene *InlineShape*, welche im Bereich dieser Textmarke liegt.

Da einen *InlineShape* textgebunden ist, also auf der selben Ebene wie Text liegt, können Sie dieses Objekt weitgehend wie Text behandeln, Konsequenterweise verfügt eine *InlineShape* auch über die *Range*-Eigenschaft. Im folgenden Beispiel wird die erste *InlineShape* im Hauptteil des Dokumentes, verborgen formatiert.

```
Sub InlineShapeAnsprechen()
  ActiveDocument.InlineShapes(1).Range.Font.Hidden = True
End Sub
```

(Eine InlineShape wird verborgen bzw. ausgeblendet.)

MS Word vergibt einer *InlineShape* keinen Namen. Sie können aber dem Bereich, welche eine InlineShape beinhaltet, eine Textmarke vergeben. Im folgenden Beispiel wird jene *InlineShape* angesprochen, welche im Textmarkenfeld mit dem Namen **MeineGrafik** eingeschlossen ist.

```
Sub InlineShapeUeberBenutzerdefinierteTextmarkeAnsprechen()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  If oDoc.Bookmarks.Exists("MeineGrafik") Then
    oDoc.Bookmarks("MeineGrafik").Range.InlineShapes(1).PictureFormat.Contrast = 0.33
  End If
End Sub
```

(Eine mit einem Textmarkenfeld versehene InlineShape wird angesprochen.)

Zusätzliche Informationen und Beispiele zum Arbeiten mit *InlineShapes* erhalten Sie im folgenden Skript dieser Reihe:

► VBA-technischer Umgang mit «Shapes» und «InlineShapes»

Das Comment-Objekt

▲ Zurück zur Übersicht der Objekte

Ein *Comment*-Objekt entsteht dann, wenn Sie vom Word-Hauptmenü den Befehl *Einfügen - Kommentar* wählen.

Dabei passieren zwei Dinge. Zum einen wird an der Stelle der Einfügemarke eine Kennzeichnung eingefügt. Gleichzeitig öffnet sich der Kommentarbereich, in dem Sie den eigentlichen Kommentar eingeben können. Die Kennzeichnung kommt im Hauptteil des Dokumentes zu liegen, der Kommentar selbst stelle eine andere Ebene (*StoryRange*) des Dokumentes dar. Beide Bereiche (*Ranges*) können Sie problemlos mit Word-VBA ansprechen.

Im folgenden Beispiel wird der Text, welcher zum ersten Kommentar im Hauptteil des Dokumentes gehört, in einer *Meldungs-Box* ausgegeben. Beachten Sie, dass Sie an Stelle der *Range*-Eigenschaft, die *Scope*-Eigenschaft benutzen müssen. Dies ist deshalb so, weil die *Range*-Eigenschaft den Kommentar an sich zurückgibt. Lassen Sie sich davon aber nicht irritieren, denn auch die *Scope*-Eigenschaft bringt einen veritablen *Range* zurück.

```
Sub KommentarKennzeichnungAuslesen()
  MsgBox ActiveDocument.Comments(1).Scope.Text
End Sub
```

(Der Text, welcher zum ersten Kommentar im Hauptteil des Dokumentes gehört, wird ausgegeben.)

Im folgenden Beispiel wird der erste Kommentar im Hauptteil des Dokumentes ausgelesen.

```
Sub KommentarAuslesen()
  MsgBox ActiveDocument.Comments(1).Range.Text
End Sub
```

(Der Text, des ersten Kommentars im Hauptteil des Dokumentes wird ausgelesen.)

Kommentare in formatierten Text umwandeln

▲ Zurück zu den Aufgabenstellungen

Wenn Sie eine Dokument, welches Kommentare enthält drucken und dabei die Option *Extras - Optionen - Drucken - Mit Dokument ausdrucken - Kommentare* aktivieren, dann werden am Schluss des Dokumentes die Kommentare ausgedruckt.

Diese Darstellungsweise ist in der Praxis nicht immer das Gelbe vom Ei. Mit dem folgenden VBA-Code, werden die Kommentare durch formatierten Text ersetzt. Das hat zum Ergebnis, dass der Ausdruck ziemlich genau so aussieht, wie sich die Sache auf dem Bildschirm präsentiert.

Erstellen Sie zuerst eine Kopie Ihres Dokumentes und testen Sie dann den folgenden VBA-Code. Das Beispiel verdeutlicht noch einmal, weshalb ein Kommentar zwei Bereiche hat, nämlich die *Scope*-Eigenschaft und die *Range*-Eigenschaft.

```
Sub KommentareInFormatiertenTextumwandeln()
  Dim Kommentar As Comment, oRange As Range
  For Each Kommentar In ActiveDocument.Comments
    pos = Kommentar.Scope.End
    Kommentar.Scope.Text = Kommentar.Scope.Text & " " & Kommentar.Range.Text
    Set oRange = ActiveDocument.Range(Start:=pos + 1, _
       End:=pos + Len(Kommentar.Range.Text) + 1)
    Kommentar.Delete
    oRange.Font.Italic = True
    oRange.Font.ColorIndex = wdGreen
  Next
End Sub
```

(Die Kommentare im Dokument werden aufgelöst und in formatierten Text, welcher gleich hinter der Kommentarreferenz zu liegen kommt, ersetzt.)

Kommentare in Fussnoten umwandeln

▲ Zurück zu den Aufgabenstellungen

Im letzten Beispiel dieses Kapitels, werden sämtliche Kommentare im Hauptteil des Dokumentes in Fussnoten umgewandelt.

```
Sub KommentareInFussnotenUmwandeln()
  Dim oDoc As Document, Kommentar As Comment
  Set oDoc = ActiveDocument
  For Each Kommentar In ActiveDocument.Comments
    oDoc.Footnotes.Add Range:=Kommentar.Scope, Text:=Kommentar.Range.Text
    Kommentar.Delete
  Next
End Sub
```

(Die Kommentare im Hauptteil des Dokumentes werden aufgelöst und mit Fussnoten ersetzt.)

Das Footnote-Objekt

▲ Zurück zur Übersicht der Objekte

Eine Fussnote entsteht dann, wenn Sie vom Word-Hauptmenü den Befehl *Einfügen - Fussnote - Fussnote* wählen.

Dabei passieren zwei Dinge. Zum einen wird an der Stelle der Einfügemarke die Fussnotenreferenz eingefügt. Gleichzeitig öffnet sich der Fussnotenbereich, in dem Sie den Fussnotentext eingeben können. Die Referenz kommt im Hauptteil des Dokumentes zu liegen, wogegen der Fussnotentext eine andere Ebene (*StoryRange*) des Dokumentes darstellt. Beide Bereiche (*Ranges*) können Sie problemlos mit Word-VBA ansprechen.

Im folgenden Beispiel wird ein *Range* gebildet, welchen den Bereich darstellt, in dem sich die erste Fussnotenreferenz des Dokumentes befindet. Beachten Sie dass dieser Bereich statt mit *Range* mit *Reference* angesprochen wird. Die *Range*-Eigenschaft bringt nämlich die Fussnote an sich zurück, welche sich in einer anderen Dokumentebene (*StoryRange* als der Hauptteile des Dokumentes befindet. Sie erhalten durch den Einsatz der *Reference*-Eigenschaft aber einen veritablen Range zurück.

```
Sub Fussnotenreferenzbereich()
  Dim oRange As Range
  Set oRange = ActiveDocument.Footnotes(1).Reference
  oRange.Select
End Sub
```

(Es wird ein Range gebildet, welcher den Bereich der ersten Fussnotenreferenz im Hauptteil des Dokumentes darstellt.)

Im folgenden Beispiel wird der Text der ersten Fussnote, welche sich im Hauptteil des Dokumentes befindet, zurückgegeben.

```
Sub FussnotentextAnsprechen()
  If ActiveDocument.Footnotes.Count > 0 Then
    MsgBox ActiveDocument.Footnotes(1).Range.Text
  End If
End Sub
```

(Der Fussnotentext der ersten Fussnote im Hauptteil des Dokumentes wird angesprochen.)

Tabulator vor Fussnotentext einbringen

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiel wird vor jeden Fussnotentext ein Tabulator eingefügt.

```
Sub TabulatorInJedenFussnotentextEinfuegen()
  Dim FN As Footnote
  For Each FN In ActiveDocument.Footnotes
    FN.Range.InsertBefore vbTab
  Next
End Sub
```

(Vor jeden Fussnotentext wird ein Tabulator eingefügt.)

Im nächsten Beispiel, wird der gesamte Fussnotentext (also der Text aller Fussnoten) in den Hauptteil eines neuen Dokumentes kopiert. Die Formatierung des Textnotentextes bleibt dabei erhalten. Die Nummerierung der Fussnoten gerät allerdings durcheinander, weil im neuen Dokument die entsprechenden Referenzen fehlen.

```
Sub FussnotentextInNeuesDokument()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add(Template:=oDoc.AttachedTemplate.FullName)
  nDoc.Content.FormattedText = oDoc.StoryRanges(wdFootnotesStory).FormattedText
End Sub
```

(Der Fussnotentext aller Fussnoten wird ein neues Dokument kopiert. Dabei bleibt die Formatierung des Textes erhalten. Die Nummerierung der Fussnoten gerät allerdings durcheinander.)

Fussnotenverzeichnis erstellen

▲ Zurück zu den Aufgabenstellungen

Im letzten Beispiel dieses Kapitels wird ein Inhaltsverzeichnis der Fussnoten erstellt. Am Schluss des Dokumentes werden nochmals alle Fussnoten aufgeführt inklusive Seitenangabe und Referenznummer der Fussnote. Dieses Verzeichnis wird mit einer Formatvorlage mit der Bezeichnung **FNIndex** hübsch formatiert. Sollte diese Formatvorlage noch nicht existieren, wird sie von der Routine *on-the-fly* erstellt.

```
Sub FussnotenIndex()
  Dim oDoc As Document, oRange As Range, FN As Footnote, FV As Style
  Set oDoc = ActiveDocument
  If oDoc.Footnotes.Count = 0 Then
    MsgBox "Das Dokument enthält keine Fussnoten."
    Exit Sub
  End If
  oDoc.Range.InsertAfter vbCrLf
  Set oRange = oDoc.Range
  Set oRange = oDoc.Range(oRange.End - 1, oRange.End - 1)
  Start = oDoc.Range.End
  For Each FN In ActiveDocument.Footnotes
    oRange.InsertAfter vbTab & FN.Index
    oDoc.Paragraphs.Last.Range.Style = wdStyleFootnoteReference
    oRange.InsertAfter vbTab & FN.Range.Text & vbTab & _
       FN.Reference.Information(wdActiveEndAdjustedPageNumber) & vbCrLf
  Next
  flag = False
  For Each FV In oDoc.Styles
    If FV.NameLocal = "FNIndex" Then
      flag = True
      Exit For
    End If
  Next
  If flag = False Then
    Set FV = oDoc.Styles.Add(Name:="FNIndex")
    FV.BaseStyle = wdStyleFootnoteText
    With oDoc.Sections(oRange.Information(wdActiveEndSectionNumber)).PageSetup
      pos = .PageWidth - .LeftMargin - .RightMargin
    End With
    With FV.ParagraphFormat.TabStops
      .Add Position:=CentimetersToPoints(0.4), Alignment:=wdAlignTabRight
      .Add Position:=CentimetersToPoints(0.5)
      .Add Position:=pos, Alignment:=wdAlignTabRight, Leader:=wdTabLeaderLines
    End With
  End If
  oDoc.Range(Start, oDoc.Range.End).Style = "FNIndex"
  oDoc.Paragraphs.Last.Range.Style = wdStyleNormal
  oDoc.Paragraphs.Last.Range.Font.Reset
End Sub
```

(Am Schluss des Dokumentes wird ein Verzeichnis der Fussnoten erstellt, welche im Hauptteil des Dokumentes vorkommen. Das Verzeichnis beinhaltet sowohl Referenznummer und Seitenangabe der Fussnoten.)

Das Endnote-Objekt

▲ Zurück zur Übersicht der Objekte

Eine *Endnote* entsteht dann, wenn Sie vom Word-Hauptmenü den Befehl *Einfügen - Fussnote - Endnote* wählen. Dabei passieren zwei Dinge. Zum einen wird an der Stelle der Einfügemarke die Endnotenreferenz eingefügt. Gleichzeitig öffnet sich der Endnotenbereich, in dem Sie den Endnotentext eingeben können. Die Referenz kommt im Hauptteil des Dokumentes zu liegen, wogegen der Endnotentext eine andere Ebene (*StoryRange*) des Dokumentes darstellt.

Der Unterschied von Fuss- und Endnoten liegt übrigens darin, dass der Fussnotentext unten auf der entsprechenden Seite, welche die Referenz enthält steht, der Endnotentext dagegen am Ende des Dokumentes.

Im folgenden Beispiel wird ein *Range* gebildet, welchen den Bereich darstellt, in dem sich die erste Endnotenreferenz des Dokumentes befindet. Beachten Sie, dass sie dazu die *Reference*-Eigenschaft verwenden müssen. Die *Range*-Eigenschaft gibt es auch zielt allerdings auf den Bereich des Endnotentextes.

```
Sub Endnotenreferenzbereich()
  If ActiveDocument.Endnotes.Count > 0 Then
    Dim oRange As Range
    Set oRange = ActiveDocument.Endnotes(1).Reference
    oRange.Select
  End If
End Sub
```

(Es wird ein Range gebildet, welcher den Bereich der Endnotenreferenz darstellt.)

Im folgenden Beispiel wird der Text der ersten Endnote im Hauptteil des Dokumentes zurückgegeben.

```
Sub EndnotentextAnsprechen()
  If ActiveDocument.Endnotes.Count > 0 Then
    MsgBox ActiveDocument.Endnotes(1).Range.Text
  End If
End Sub
```

(Der Endnotentext der ersten Endnote im Hauptteil des Dokumentes wird angesprochen.)

Im letzten Beispiels dieses Kapitels werden die Kommentare im Hauptteil des Dokumentes aufgehoben und in Endnoten umgewandelt.

```
Sub KommentareInEndnotenUmwandeln()
  Dim oDoc As Document, Kommentar As Comment
  Set oDoc = ActiveDocument
  For Each Kommentar In ActiveDocument.Comments
    oDoc.Endnotes.Add Range:=Kommentar.Scope, Text:=Kommentar.Range.Text
    Kommentar.Delete
  Next
End Sub
```

(Die Kommentare im Hauptteil des Dokumentes werden aufgelöst und durch Endnoten ersetzt.)

Das Field-Objekt

▲ Zurück zur Übersicht der Objekte

Ein Feld entsteht z.B. dann, wenn sie vom Word-Hauptmenü den Befehl *Einfügen - Feld...* verwenden.

In unserem ersten Beispiel wird der Feldcode des ersten Feldes im Hauptteil des Dokumentes zurückgegeben. Beachten Sie dabei, dass Sie dafür die Eigenschaft *Code* verwenden müssen. Sie erhalten allerdings einen veritablen *Range* zurück, auch wenn der Begriff anders lautet.

```
Sub FeldcodeZurueckgeben()
  MsgBox ActiveDocument.Fields(1).Code.Text
End Sub
```

(Der Feldcode des ersten Feldes im Hauptteil des Dokumentes wird zurückgegeben.)

Im nächsten Beispiel wird das Ergebnis des ersten Feldes im Hauptteil des Dokumentes zurückgegeben. Beachten Sie dabei, dass Sie dafür die Eigenschaft *Result* verwenden müssen. Sie erhalten allerdings einen veritablen *Range* zurück, auch wenn der Begriff anders lautet.

```
Sub FeldergebnisZurueckgeben()
  MsgBox ActiveDocument.Fields(1).Result.Text
End Sub
```

(Das Ergebnis des ersten Feldes im Hauptteil des Dokumentes wird zurückgegeben.)

Das FormField-Objekt

▲ Zurück zur Übersicht der Objekte

Ein konventionelles Formularfeld entsteht dann, wenn Sie über die Symbolleiste mit der Bezeichnung *Formular* z.B. ein Textformularfeld in das Dokument einbringen.

Im ersten Beispiel wird ein *Range* gebildet, welcher den Bereich des Formulartextfeldes mit dem Namen **Text1** umfasst.

```
Sub RangeEinesFormularfeldesAnsprechen()
  Dim oRange As Range
  Set oRange = ActiveDocument.FormFields("Text1").Range
End Sub
```

(Der Bereich des Formularfeldes mit der Bezeichnung Text1 wird angesprochen.)

Verwenden Sie die *Result*-Eigenschaft, um den Inhalt eines Formularfeldes auszulesen oder zu setzen.

```
Sub ErgebnisEinesFormularfeldesZurueckgeben()
  MsgBox ActiveDocument.FormFields("Text1").Result
End Sub
```

(Der Text (Inhalt) eines konventionellen Formulartextfeldes wird zurückgegeben.)

Das Table-Objekt

▲ Zurück zur Übersicht der Objekte

Word-Tabellen werden sehr gerne eingesetzt und haben bei den Benutzern einen hohen Stellenwert. Das Arbeiten mit Word-Tabellen ist vielseitig, konsistent und nicht sehr schwierig.

Im folgenden Beispiel werden die Zellen einer Word-Tabelle sequentiell abgearbeitet. In jede der Zellen wird eine fortlaufende Nummer eingefügt. Die Reihenfolge, wie die Zellen bedient werden, entspricht der Leserichtung, also von links mach rechts und von oben nach unten.

```
i = 0
Dim Zelle as Cell
For Each Zelle In oDoc.Tables(1).Range.Cells
  i = i + 1
  Zelle.Range.Text = i
Next
```

(Die Zellen der ersten Word-Tabelle im Dokument werden sequentiell abgearbeitet.)

Bei Word-Tabellen handelt es sich nicht nur um ein sehr populäres sondern auch um ein ziemlich umfangreiches Gebiet. Deshalb wird dieses Thema in einem eigenen Skript dieser Reihe abgehandelt. Sie finden dieses Skript unter dem folgenden Link:

► VBA-technischer Umgang mit Word-Tabellen

Das Cell-Objekt

▲ Zurück zur Übersicht der Objekte

Mit dem *Cell*-Objekt sind Zellen von Tabellen gemeint. Das folgende Beispiel gibt den Inhalt der ersten Zelle in der ersten Tabelle des Hauptteils des aktiven Dokumentes zurück. Beachten Sie, dass eine Tabellenzelle (auch eine leere) immer mit zwei Steuerzeichen endet. Diese beiden Zeichen sollten Sie konsequenterweise abtrennen, wie dies im folgenden Beispiel gezeigt wird.

```
Sub TabellenzelleAuslesen()
  strText = ActiveDocument.Tables(1).Cell(1, 1).Range.Text
  strText = Left(strText, Len(strText) - 2)
  MsgBox strText
End Sub
```

(Der Inhalt einer Tabellenzelle wird ausgelesen.)

Tabellenspalte durchsuchen

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiel wird in der dritten Spalte der ersten Tabelle im Hauptteil des Dokumentes der Ortsbegriff Basel durch das französischsprachige Equivalent *Bâle* ersetzt. Da die Zellen einer Spalte keinen zusammenhängenden Bereich darstellen (wenn man es in der üblichen Leserichtung betrachtet), behandeln wir die Spalte, als eine Sammlung von Zellen, welche wir mit einer Schleife bedienen können. Das Objekt Columns gibt es zwar in Word-VBA, allerdings verfügt es nicht über die *Range*-Eigenschaft. Diese Tatsache sollte uns nicht mehr erstaunen, denn eine Spalte hat im Sinne des Word-Objekt-Models kein klarer Start- bzw. Endpunkt; sie liegt quer im Felde. Würde man die Spalte in Leserichtung anmalen, müsste der Pinsel immer wieder von neuem angesetzt bzw. abgesetzt werden.

```
Sub TabellenspalteDurchsuchen()
  Application.ScreenUpdating = False
  Dim oTable As Table, Zelle As Cell
  Set oTable = ActiveDocument.Tables(1)
  For Each Zelle In oTable.Columns(3).Cells
    With Zelle.Range.Find
      .Text = "Basel"
      .Replacement.Text = "Bâle"
      .Execute Replace:=wdReplaceAll
    End With
  Next
  Application.ScreenUpdating = True
End Sub
```

(Die Routine demonstriert, wie eine Tabellenspalte durchsucht werdern kann.)

Sehr ausführliche Informationen über das Arbeiten mit Tabellenzellen und Word-Tabellen ganz allgemein und zusätzliche Beispiele, erhalten Sie im folgenden Skript dieser Reihe:

► VBA-technischer Umgang mit Word-Tabellen

Das Find-Objekt

▲ Zurück zur Übersicht der Objekte

Das *Find*-Objekt wird immer dann eingesetzt, wenn nach einem vorgegebenen Text, nach bestimmten Steuerzeichen oder nach einer bestimmten Formatierung im Dokument gesucht werden soll. Nicht nur die Suche ist dabei unterstützt, sondern auch der systematische Austausch. Das *Find*-Objekt kann sowohl auf das *Selection*-Objekt wie auch auf das *Range*-Objekt angewendet werden.

Als einzige grosse Ausnahme soll der Einsatz über das *Selection*-Objekt performander sein, als der Gebrauch über das Range-Objekt. Versuche bestätigen dies, wenn der Unterschied auch nicht besonders krass ausfällt (plus 10 - 15% zu Ungunsten des *Range*-Objektes). Wieso dem so ist, weiss ich leider nicht.

Worthäufigkeit ermitteln

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiel wird **`Selection.Find`** eingesetzt, um zu zählen, wie oft ein vorgegebenes Wort im Hauptteil des Dokumentes vorkommt. Das erste Beispiel arbeitet also mit der *Selection*- und nicht mit der *Range*-Eigenschaft.

```
Sub SuchenMitDemSelectionObjekt()
  Application.ScreenUpdating = False 'steigert Performanz/dämmt das Flackern ein
  Dim oRange As Range
  Set oRange = Selection.Range 'Aktuelle Position der Einfügemarke sichern
  ActiveDocument.Range(0, 0).Select 'Am Beginn des Dokumentes starten
  Anz = 0
  With Selection.Find
    .ClearFormatting
    .Text = InputBox("Gegeben Sie ein Wort ein:")
    .Execute
    While .Found = True
      Anz = Anz + 1
      .Execute
    Wend
  End With
  MsgBox "Das Wort erscheint " & Anz & " mal im Hauptteil des Dokumentes"
  oRange.Select 'Position der Einfügemarke wie am Anfang
  Application.ScreenUpdating = True
End Sub
```

(Die Routine zählt, wie oft ein vorgegebenes Wort im Hauptteil des Dokumentes auftritt. Die Aufgabe wird in diesem Beispiel mit dem Selection-Objekt gelöst. Normalerweise ist das Arbeiten mit Range-Objekten performander. Die Find-Methode bildet da die einzig, mir bekannte Ausnahme.)

Das nächste Beispiel macht funktionell das selbe, arbeitet allerdings mit der *Range*-Eigenschaft. Es ist, wie schon erwähnt, etwa 10% weniger performand, wie das vorhergehende Beispiel, kommt aber bei gleichem Komfort mit weniger Code aus.

```
Sub SuchenMitDemRangeObjekt()
  Application.ScreenUpdating = False 'steigert Performanz/dämmt flackern ein
  Anz = 0
  With ActiveDocument.Content.Find
    .ClearFormatting
    .Text = InputBox("Gegeben Sie ein Wort ein:")
    .Execute
    While .Found = True
      Anz = Anz + 1
      .Execute
    Wend
  End With
  MsgBox "Das Wort erscheint " & Anz & " mal im Hauptteil des Dokumentes"
  Application.ScreenUpdating = True
End Sub
```

(Die selbe Aufgabenstellung lässt sich auch über die Range-Eigenschaft statt der Selection-Eigenschaft lösen. Verblüffend ist, dass die Range-Methode hier etwas unperformander ist. Weniger Code benötigt sie, wenn Sie die Range-Eigenschaft einsetzen, allerdings alle weil.)

Ich persönlich bevorzuge trotzdem die *Range*-Methode, das diese Mimik übersichtlicher ist, sich als ausbaufähiger erweist (siehe nächstes Beispiel) und mit weniger Code auskommt.

Im folgenden Beispiel wird ebenfalls gezählt, wie oft ein vorgegebenes Wort im Dokument auftritt. Diesmal werden aber alle Dokumententeile berücksichtig. Falls Sie nur in einigen Teilen des Dokumentes das Auftreten zählen möchten, dann können Sie diese Routine auf sehr einfache Art und Weise dahingehend anpassen.

Der Code arbeitet mit dem *StoryRange*-Objekt. Dieses Objekt wird in einem der nächsten Kapitel noch ausführlich beschrieben.

```
Sub WortZählenInAlleDokumententeilen()
  Application.ScreenUpdating = False
  Titel = "Wörter zählen"
  strIn = InputBox("Nach welchem Wort soll gesucht werden?", Titel)
  If strIn = "" Then Exit Sub
  Dim oStory As Range
  Anz = 0
  For Each oStory In ActiveDocument.StoryRanges
    Anz = Anz + WortSuchen(oStory, strIn)
    While Not (oStory.NextStoryRange Is Nothing)
      Set oStory = oStory.NextStoryRange
      Anz = Anz + WortSuchen(oStory, strIn)
    Wend
  Next
  Application.ScreenUpdating = True
  Q = Chr(34)
  strT = "Für das Wort " & Q & strIn & Q & " wurden " & _
     Anz & " Entsprechungen gefunden."
  MsgBox strT, vbInformation, Titel
End Sub

Private Function WortSuchen(oStory, ByVal Wort As String) As Long
  Flag = False
  Anzahl = 0
  Select Case oStory.StoryType
    Case wdMainTextStory: Flag = True
    Case wdFootnotesStory: Flag = True
    Case wdEndnotesStory: Flag = True
    Case wdCommentsStory: Flag = True
    Case wdTextFrameStory: Flag = True
    Case wdPrimaryHeaderStory: Flag = True
    Case wdFirstPageHeaderStory: Flag = True
    Case wdEvenPagesHeaderStory: Flag = True
    Case wdPrimaryFooterStory: Flag = True
    Case wdFirstPageFooterStory: Flag = True
    Case wdEvenPagesFooterStory: Flag = True
  End Select
  If Flag = False Then Exit Function
  With oStory.Find
    .ClearFormatting
    .Text = Wort
    .Execute
    While .Found = True
      Anzahl = Anzahl + 1
      .Execute
    Wend
  End With
  WortSuchen = Anzahl
End Function
```

(Auch diese Routine zählt, wie oft eine vorgegebenes Wort im Dokument auftritt. Diesmal wird aber nicht nur der Hauptteil des Dokumentes berücksichtigt, sondern alle Dokumententeile, welche Sie im Code einschalten. So wie der Code hier vorliegt, werden alle Teile in die Zählung mit einbezogen.)

Im folgenden wir gezählt wie viele rot-formatierte Zeichen im Hauptteil des Dokumentes vorhanden sind.

```
Sub RoteZeichenZaehlen()
  Anz = 0
  With ActiveDocument.Content.Find
    .ClearFormatting
    .Text = ""
    .Format = True
    .Font.ColorIndex = wdRed
    .Execute
    While .Found = True
      Anz = Anz + Len(.Parent.Text)
      .Execute
    Wend
  End With
  MsgBox "Die Zusatzzahl lautet:" & vbCrLf & Anz
End Sub
```

(Hier wird nicht nach dem Vorkommen einer Zeichenfolge geforscht, sondern nach einer bestimmten Formatierung.)

Tabellenspalte bearbeiten

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiel wird in der dritten Spalte der ersten Tabelle im Hauptteil des Dokumentes der Ortsbegriff *Basel* durch das französischsprachige Equivalent *Bâle* ersetzt. Da die Zellen einer Spalte keinen zusammenhängenden Bereich darstellen (wenn man es in der üblichen Leserichtung betrachtet), behandeln wir die Spalte, als eine Sammlung von Zellen, welche wir mit einer Schleife bedienen können. Das Objekt *Columns* gibt es zwar in Word-VBA, allerdings verfügt es nicht über die *Range*-Eigenschaft. Diese Tatsache sollte uns nicht mehr erstaunen, denn eine Spalte hat im Sinne des Word-Objekt-Models kein klarer Start- bzw. Endpunkt; sie liegt quer im Felde. Würde man die Spalte in Leserichtung anmalen, müsste der Pinsel immer wieder von neuem angesetzt bzw. abgesetzt werden.

```
Sub TabellenspalteDurchsuchen()
  Application.ScreenUpdating = False
  Dim oTable As Table, Zelle As Cell
  Set oTable = ActiveDocument.Tables(1)
  For Each Zelle In oTable.Columns(3).Cells
    With Zelle.Range.Find
      .Text = "Basel"
      .Replacement.Text = "Bâle"
      .Execute Replace:=wdReplaceAll
    End With
  Next
  Application.ScreenUpdating = True
End Sub
```

(Die Routine demonstriert, wie eine Tabellenspalte durchsucht werdern kann.)

Das Frame-Objekt

▲ Zurück zur Übersicht der Objekte

Beim Frame-Objekt handelt es sich um einen Positionsrahmen. Das Objekt entsteht z.B. dann, wenn Sie über die Symbolleiste mit der Bezeichnung *Formular* die Schaltfläche Positionsrahmen einfügen betätigen.

Positionsrahmen sind mit den neueren Word-Versionen (ab Word 97) etwas aus der Mode gekommen. Als Ersatz sieht Microsoft das Textfeld vor, welches u.a. bessere Formatierungsmöglichkeiten anbietet. Positionsrahmen haben aber auch ihre Stärke, so können diese doch z.B. Bestandteile einer Formatvorlage sein.

Im folgenden Beispiel wird ein Objekt gebildet, welches den Bereich des ersten Positionsrahmen im Hauptteil des Dokumentes umfasst.

```
Sub RangeObjektDesBereichesEinesPositionsrahmenBilden()
  Dim oRange As Range
  Set oRange = ActiveDocument.Frames(1).Range
End Sub
```

(Es wird ein Range gebildet, welcher den Bereich des ersten Positionsrahmen im Hauptteil des Dokumentes umfasst.)

Im folgenden Beispiel wird der Text des ersten Positionsrahmens im Hauptteil des Dokumentes zurück gegeben.

```
Sub TextEinesPositionsrahmensZurueckgeben()
  MsgBox ActiveDocument.Frames(1).Range.Text
End Sub
```

(Der Text des ersten Positionsrahmen im Hauptteil des Dokumentes wird zurück gegeben.)

Das SpellingError-Objekt (Rechtschreibfehler)

▲ Zurück zur Übersicht der Objekte

Bereiche, welche die Rechtschreibroutine von MS Word als fehlerhaft einstuft, stellen eine Sammlung von *Range*-Objekten dar. Das *SpellingError*-Objekt gibt also eine *Range*-Objekt zurück.

Im folgenden Beispiel wird der erste Rechtschreibfehler im Hauptteil des Dokumentes zurückgegeben.

```
Sub RechtschreibfehlerZurueckgeben()
  MsgBox ActiveDocument.SpellingErrors(1).Text
End Sub
```

(Der erste Rechtschreibfehler im Hauptteil des Dokumentes wird zurückgegeben..)

Rechtschreibfehler automatisiert aufnehmen

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiel werden alle Dokumente, welche sich im Verzeichnis mit dem Namen **C:\Ablage** befinden nach Rechtschreibfehlern untersucht. Die Wörter, welche MS Word in diesen Dokumenten moniert werden dann an das Benutzerwörterbuch mit dem Namen Text.dic angehängt. Sollte dieses Wörterbuch noch nicht existieren, wird es erstellt.

Doppelte Einträge im Benutzerwörterbuch und die fehlende Sortierung der Wörter stört MS Word bei der Rechtschreibprüfung übrigens nicht. Word bereinigt diese Mängel früher oder später aus eigenem Antrieb.

Kopieren Sie den folgenden Code im VBA-Editor in ein neues Modul, passen Sie im Kopf des Codes die beiden Konstanten mit den Namen Verzeichnis und Filter Ihren persönlichen Bedürfnissen an und testen Sie dann die Lösung.

```
Private Const Verzeichnis = "C:\Ablage"
Private Const Filter = "*.doc"

Sub RechtschreibfehlerNachBeunutzerDic()
  Pfad = Options.DefaultFilePath(wdProofingToolsPath) & "\Test.Dic"
  Close
  Open Pfad For Append As #1
  Dim oDoc As Document, oRange As Range
  tmp = Dir(Verzeichnis & "\" & Filter)
  While Not tmp = ""
    On Error Resume Next
    Set oDoc = Documents.Open(tmp, AddToRecentFiles:=False, _
       ConfirmConversions:=False)
    If Err.Number > 0 Then
      oDoc.Close SaveChanges:=False
    End If
    On Error GoTo 0
    StatusBar = "Das Dokument " & tmp & " wird jetzt verarbeitet."
    DoEvents
    For Each oRange In ActiveDocument.SpellingErrors
      Print #1, oRange.Text
    Next
    oDoc.Close SaveChanges:=False
    tmp = Dir
  Wend
  Close #1
  Q = Chr(34)
  Shell "notepad.exe " & Q & Pfad & Q, vbNormalFocus
End Sub
```

(Wörter in einer Sammlung von Dokumenten, welche der Rechtschreibprüfung von MS Word unbekannt sind, werden automatisiert erfasst und dann in ein Benutzerwörterbuch eingetragen.)

Das StoryRange-Objekt

▲ Zurück zur Übersicht der Objekte

Das *StoryRange*-Objekt spricht einen bestimmten Dokumententeil in einer Word-Datei an. Sollten Sie sich unter einem *StoryRange* noch nichts Konkretes vorstellen können, dann lesen Sie bitte das Kapitel **Was ist ein Range** am Anfang dieses Skriptes. Beachten Sie auch die Tabelle etwas weiter unten, die alle elf **StoryTypes**, welche das Word-Objekt-Modell definiert, auflistet.

Der Hauptteil des Dokumentes heisst übrigens **`wdMainTextStory`**. Dafür stellt Word aber eine eine handlichere Alternative bereit, handelt es sich dabei doch um eine synonym für den vertrauten Ausdruck **`ActiveDocument`**. Somit tätigen die beiden folgenden Code-Beispiele das selbe, sie kopieren nämlich den Text des Hauptteils des Dokumentes in ein neues Dokument, wobei die Formatierung erhalten bleibt.

```
Sub HauptteilDesDokumentesKopieren1()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add(Template:=oDoc.AttachedTemplate.FullName)
  nDoc.StoryRanges(wdMainTextStory).FormattedText = _
     oDoc.StoryRanges(wdMainTextStory).FormattedText
End Sub
```

(Dieses Beispiel, welches den Hauptteil des Dokumentes in ein neues Dokument kopiert, hat wohl nur demonstrativen Charakter. Verwenden Sie in der Praxis das nächste Beispiel, welches einfacher oder zu mindest populärer ist.)

```
Sub HauptteilDesDokumentesKopieren2()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add(Template:=oDoc.AttachedTemplate.FullName)
  nDoc.Range.FormattedText = oDoc.Range.FormattedText
End Sub
```

(Die beschriebene Aufgabenstellung lässt sich in diesem einfachen Fall auch ohne StoryRange-Objekt lösen.)

Schon wesentlich interessanter ist das folgende Beispiel. Dabei wird der Fussnotenteil des aktiven Dokumentes in ein neues Dokument kopiert.

```
Sub FussnotentextInNeuesDokument()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add(Template:=oDoc.AttachedTemplate.FullName)
  If oDoc.Footnotes.Count > 0 Then
    nDoc.Content.FormattedText = _
       oDoc.StoryRanges(wdFootnotesStory).FormattedText
  End If
End Sub
```

(Der Fussnotentext aller Fussnoten wird ein neues Dokument kopiert. Dabei bleibt die Formatierung des Textes erhalten. Die Nummerierung der Fussnoten gerät allerdings durcheinander, da die Referenzen im neuen Dokument fehlen)

Bleiben noch zwei wichtige Punkte zu klären. Der erste Punkt beschäftigt sich mit der Frage, wie man programmtechnisch feststellen kann, ob ein bestimmter StoryRange im Dokument existiert und der zweite Punkt - vielleicht noch wichtiger - beschäftigt sich mit der Frage wie alle StoryRanges im Dokument sequentiell abgearbeitet werden können, wenn gar nicht genau bekannt ist, welche und wie viele dieser Stories im Dokument vorhanden sind.

Feststellen, ob eine bestimmte *Story* im Dokument vorhanden ist
Leider gibt es dafür keine direkte Abfrage. Sie müssen sich hier indirekt vorgehen oder sich mit einer Schleife behelfen. In den folgenden zwei Beispielen wird eruiert, ob das aktive Dokument die **`wdEvenPagesHeaderStory`** enthält.

```
Sub EruierenObErsteSeiteStoryVorhanden1()
  If ActiveDocument.Sections(1).PageSetup.DifferentFirstPageHeaderFooter = True Then
    MsgBox "ErsteSeiteStory vorhanden"
  Else
    MsgBox "ErsteSeiteStory nicht vorhanden"
  End If
End Sub
```

(Ermittelt indirekt, ob der StoryRange für den Bereich Erste Seite anders vorhanden ist.)

```
Sub EruierenObErsteSeiteStoryVorhanden2()
  Dim oStory As Range
  Flag = False
  For Each oStory In ActiveDocument.StoryRanges
    If oStory.StoryType = wdFirstPageHeaderStory Then
      Flag = True
      Exit For
    End If
  Next
  If Flag = True Then
     MsgBox "ErsteSeiteStory vorhanden"
  Else
    MsgBox "ErsteSeiteStory nicht vorhanden"
  End If
End Sub
```

(Ermittelt mittels Schleife, ob der StoryRange für den Bereich Erste Seite anders vorhanden ist.)

Die beiden besprochenen Beispiele haben übrigens nicht in jedem Falle die selbe Wirkung. Die erste Mimik stellt ein Potential fest, wogegen die zweite Mimik das tatsächliche Vorhandensein eruiert.

Im folgenden Beispiel wird - ebenfalls auf indirekte Art und Weise ermittelt - ob der *StoryRange* mit der Bezeichnung **`wdFootnotesStory`** im aktiven Dokument vorhanden ist.

```
Sub FeststellenObFussnotenteilVorhanden()
  If ActiveDocument.Footnotes.Count > 0 Then
    MsgBox "Es ist ein Fussnotenteile vorhanden."
  Else
    MsgBox "Es ist kein Fussnotenteile vorhanden."
  End If
End Sub
```

(Stellt - indirekt - fest, ob ein Fussnotenteil vorhanden ist.)

Alle *StoryRanges* eines Dokumentes sequentiell abarbeiten

▲ Zurück zum Anfang

Oft ist nicht bekannt, welche Dokumententeile in einem Dokument wirklich enthalten sind und z.B. im Falle von Textfeldern, welche Anzahl eines bestimmten *StoryType* vorliegt.

Stellen Sie sich einmal vor, Sie stehen vor der praxisorientierten Aufgabe in einem Word-Dokument alle Felder zu aktualisieren. Das scheint bei einem flüchtigen Blick einfach lösbar zu sein, braucht es dazu doch nur eine einzige Word-VBA Codezeile nämlich: **`ActiveDocument.Fields.Update`**. Unser erster Versuch sieht also wie folgt aus:

```
Sub FelderAktualisieren1()
  ActiveDocument.Fields.Update
End Sub
```

(Aktualisiert die Felder im Hauptteil des Dokumentes. Alle anderen Teile des Dokumentes sind darin nicht eingeschlossen.)

Die Sache hat aber einen Haken, denn dieser Befehl wirkt sich ausschliesslich auf den Hauptteil des Dokumentes aus. Die folgende Lösung kommt unserem Ziel bereits schon wesentlicher näher, wenn es auch immer noch gewichtige Einschränkungen gibt, auf welche hier noch eingegangen werden muss

```
Sub FelderAktualisieren2()
  Dim oStory As Range
  For Each oStory In ActiveDocument.StoryRanges
    oStory.Fields.Update
  Next
End Sub
```

(Eine bequeme Art StoryRanges in MS Word sequentiell abzuarbeiten. Leider funktioniert das so nicht korrekt in allen Fällen, welche denkbar sind und welche in der Praxis auch vorkommen.)

Ob es sich nun um ein weiteres Fehlverhalten von MS Word handelt, oder ob die Sache an der fehlenden Rekursivität der Mimik scheitert - ich weiss es nicht. Allenfalls ist diese Wertung Ansichtsache, praxisgerecht ist es jedenfalls bestimmt nicht.

Das Problem beginnt mit Textfeldern. Beim Versuch, die *StoryRanges* in einem Dokument sequentiell abzuarbeiten, wird zwar - falls vorhanden - das erste Textfeld gefunden, nicht aber weitere Textfelder, falls mehrere Textfelder direkt im Hauptteil des Dokument vorkommen.

Beachten Sie, dass es in MS Word möglich ist, Textfelder zu verschachteln (also Textfeld in Textfeld). Diese Art von Rekursivität wird mit der oben abgedruckten Routine erkannt und verarbeitet. Dann ist aber Schluss und zwar unabhängig davon, ob das Dokument überhaupt verschachtelte Textfelder beinhaltet oder nicht. Die Schleife verarbeitet also nicht das nächste Textfeld, welches in der Hauptebene liegt, sondern betrachtet den gesamten Job als erledigt. Das Problem ist also, dass die Mimik zwar der Hierarchie folgt. Ist aber der kleinste Zweig erreicht, wird nicht beim nächsten Hauptzweig weiter gefahren. Die Vorstellung endet dann einfach.)-:

Die nächste Mimik ist zwar bereits etwas komplexer, liefert aber bereits bessere Resultate und ist damit allgemeingültiger und praxisgerechter.

Felder in allen Dokumententeilen aktualisieren

▲ Zurück zu den Aufgabenstellungen

```
Sub FelderAktualisieren3()
  Dim oStory As Range
  For Each oStory In ActiveDocument.StoryRanges
    oStory.Fields.Update
    While Not (oStory.NextStoryRange Is Nothing)
      Set oStory = oStory.NextStoryRange
      oStory.Fields.Update
    Wend
  Next
End Sub
```

(Diese etwas komplexere Mimik deckt schon wesentlich mehr Fälle ab und ist damit allgemeingültiger.)

Voilà, man mag nun in der oben abgedruckten Mimik noch eine programmtechnische Unschönheit sehen, was das Thema strukturiertes Programmieren betrifft, den der Befehl **`oStory.Fields.Update`** wird zweimal ausgegeben, einmal vor der Schleife und einmal in der Schleife. Das wirkt sich im Beispiel nicht nachteilig aus, denn die Funktion, welche repetitiv ausgeführt wird, erschöpft sich in einer einzigen Codezeile. Da dies in der Praxis aber nicht in immer der Fall ist, könnte man den selben Code auch wie folgt strukturieren.

```
Sub FelderAktualisieren4()
  Dim oStory As Range
  For Each oStory In ActiveDocument.StoryRanges
    StoryVerarbeiten oStory
    While Not (oStory.NextStoryRange Is Nothing)
      Set oStory = oStory.NextStoryRange
      StoryVerarbeiten oStory
    Wend
  Next
End Sub

Private Sub StoryVerarbeiten(oStory As Range)
  oStory.Fields.Update
End Sub
```

(Das selbe Beispiel nochmals, etwas besser strukturiert.)

Ich betone nochmals: das soeben aufgeführte Beispiel bring substanziell nichts Neues, es ist aber in einem gewissen Sinne besser strukturiert.

Die letzen beiden Beispiele sind wohl praktisch gesehen, dass Optimum, welches Sie erreichen können. Wegen eines sehr gewichtigen Fehlverhaltens von MS Word, werden allerdings Dokumententeile, welche folgende beiden Bedingungen erfüllen, nicht erfasst.

- Es handelt sich um ein Textfeld.
- Das Textfeld befindet sich in der Kopf- oder Fusszeile.

Näheres zu diesem folgenschweren Fehler in MS Word erfahren Sie in einem Skript von Microsoft. Dort ist auch ein sehr umständlicher Rundherumlauf publiziert. Den Link zu diesem Artikel, finden Sie am Ende dieses Skriptes.

☹ Es ist sehr zu bedauern, dass es Microsoft nicht schafft, konsistente Mimiken für eine Architektur zur Verfügung zu stellen, welche diese Weltfirma ja selbst erschaffen hat. Es scheint fast so, dass, wenn es interessant und etwas komplizierter wird, die Entwickler in Redmond selbst sang- und klanglos zu Boden gehen. Dies sollte bei einem führenden Produkt aus der Schmiede des grössten Softwareherstellers für Kleinrechner nicht geschehen. Solche Vorkommnisse werfen ein schiefes Licht auf die Produkte der Firma Microsoft.

Tabelle der Dokumententeile (StoryTypes) in MS Word

▲ Zurück zum Anfang

Im der folgenden Tabelle finden Sie die elf StoryTypes aufgelistet, welche das Word-Objekt-Modell definiert.

Komponente	Word-Objekt	Relation	Bemerkung	Beispiel
Haupttext	MainTextStory	1 pro Dokument	MainTextStory und Dokument können synonym behandelt werden.	`ActiveDocument.Fields.Update`
Hauptkopfzeile	PrimaryHeaderStory	1 pro Abschnitt (Section)	Ist immer vorhanden, kann aber auch leer sein. Kann auch der OddPageHeaderStory entsprechen, welche es nominell nicht gibt	`ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Fields.Update`
Kopfzeile für erste Seite	FirstPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Erste Seite anders" aktiviert ist.	`ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage).Range.Fields.Update`
Kopfzeile für gerade Seite	EvenPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Gerade-/ungerade Seiten anders" aktiviert ist.	`ActiveDocument.Sections(1).Headers(wdHeaderFooterEvenPages).Range.Fields.Update`
Hauptfusszeile	PrimaryFooterStory	1 pro Abschnitt (Section)	Siehe Hauptkopfzeile	`ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range.Fields.Update`
Fusszeile für erste Seite	FirstPageFooterStory	0 oder 1 pro Abschnitt (Section)	Siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterFirstPage).Range.Fields.Update`
Fusszeile für gerade Seite	EvenPageFooterStory	0 oder 1 pro Abschnitt (Section)	Siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterEvenPages).Range.Fields.Update`
Kommentare	CommentsStory	0 zu n pro Dokument	Kommentare können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Comments(1).Range.Fields.Update`
Endnote	EndnotesStory	0 oder 1 pro Dokument	Endnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Endnotes(1).Range.Fields.Update`
Fussnoten	FootnotesStory	0 zu n pro Dokument	Fussnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Footnotes(1).Range.Fields.Update`
Textfeld	TextFrameStory	0 zu n pro Story	Textfelder sind in erster Linie Objekte (Elemente) von Shapes	`ActiveDocument.Shapes(1).TextFrame.TextRange.Fields.Update`

Range-Objekte bilden

▲ Zurück zum Anfang

Wie schon einige Beispiele in diesem Skript aufzeigen, können Sie nicht nur *Ranges* ansprechen, sondern auch solche selbst bilden. Wir erinnern uns, dass ein *Range* eigentlich nichts anderes darstellt, als einen Bereich mit einem Start- und Endpunkt.

Wenn Sie einen *Range* gebildet haben (als Objekt in Ihrem Programm), können Sie Start- und End-Punkt dieses Ranges auch dahingehend manipulieren, dass entweder Start oder Ende (oder beides) verändert wird. Dazu ist die ***SetRange***-Methode zuständig.

Die Syntax der *SetRange*-Methode ist dabei nicht besonders handlich ausgefallen, muss man doch zwingend immer den Start- und End-Punkt codieren, was oft zu einer gewissen Redundanz beim Codieren führt.

Im folgenden Beispiel, wird eine Range gebildet, welches den Bereich beinhaltet, welcher gerade markiert ist. Beachten Sie, dass dieses Beispiel im Kapitel *Selection*-Objekt bereits näher erläutert wurde.

```
Sub SelectionToRange()
  Dim oRange As Range
  Set oRange = Selection.Range
End Sub
```

(Es wird ein Range-Objekt gebildet, welches die Markierung darstellt.)

Das folgende Beispiel bildet ein Range aus den ersten zehn Zeichen im Hauptteil des Dokumentes. Beachten Sie, dass die Zählung bei 0 und nicht bei 1 beginnt.

```
Sub RangeObjektErstellen()
  Dim oRange As Range
  Set oRange = ActiveDocument.Range(Start:=0, End:=9)
End Sub
```

(Aus den zehn ersten Zeichen im Dokument wird ein Range gebildet.)

Im folgenden wird ein *Range*-Objekt gebildet, welches bei der Textmarke mit dem Namen **MeinAnfang** beginnt und mit der Textmarke **MeinEnde** endet.

```
Sub TextZwischenZweiTextmarkenMarkieren()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  If oDoc.Bookmarks.Exists("MeinAnfang") And _
     oDoc.Bookmarks.Exists("MeinEnde") Then
    oDoc.Range(Start:=oDoc.Bookmarks("MeinAnfang").Range.Start, _
       End:=oDoc.Bookmarks("MeinEnde").Range.Start).Select
  End If
End Sub
```

(Bildet ein Range-Objekt, welches den Bereich zwischen zwei offenen Textmarken darstellt.)

Im folgenden wird festgestellt, um die wievielte Tabelle es sich im Dokument handelt. Die Routine prüft zuerst, ob sich die Einfügemarke überhaupt in einer Tabelle befindet.

```
Sub IndexDerAktuellenTabelleHerausfinden()
  If Not Selection.Information(wdWithInTable) Then
    MsgBox "Die Einfügemarke befindet sich nicht in einer Tabelle."
    Exit Sub
  End If
  Dim oRange As Range
  Set oRange = ActiveDocument.Range(0, Selection.Tables(1).Range.End)
  MsgBox "Das ist die " & oRange.Tables.Count & ". Tabelle im Dokument."
End Sub
```

(Stellt fest, die wievielte Tabelle im Dokument markiert ist.)

Die SetRange-Methode

Um Start und Ende eines *Range*-Objektes zu manipulieren kann die *SetRange*-Methode heran gezogen werden.

Im folgenden Beispiel geht es darum, eine komplex gestaltete Fusszeile zu erstellen ohne in die entsprechende Ansicht zu wechseln und ohne die Einfügemarke im Dokument zu bewegen. Der Code macht dabei reichlich Gebrauch von der *SetRange*-Methode.

```
Sub FusszeileZusammenbauen()
  If Application.Version = "8.0" Then 'Word 97
    strDatum = "\@" & Chr(34) & "t. MMMM jjjj" & Chr(34)
  Else 'Word2000 und höher
    strDatum = "\@" & Chr(34) & "d. MMMM yyyy" & Chr(34)
  End If
  Dim oDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.Delete 'Fusszeilenbereich löschen
  With oDoc.Sections(1).PageSetup 'Breite des bedruckbaren Bereiches
    Breite = .PageWidth - .LeftMargin - .RightMargin
  End With
  With oRange.ParagraphFormat
    .TabStops.ClearAll 'Tabstopps löschen
    .TabStops.Add Position:=0 'Tabstopp links
    .TabStops.Add Breite / 2, Alignment:=wdAlignTabCenter 'Tabstopp Mitte
    .TabStops.Add Position:=Breite, Alignment:=wdAlignTabRight 'Tabstopp rechts
  End With
  oRange.Fields.Add Range:=oRange, Type:=wdFieldAuthor 'Author
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.InsertAfter vbTab
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  oRange.Fields.Add Range:=oRange, Type:=wdFieldCreateDate, Text:=strDatum
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.InsertAfter vbTab & "Seite "
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  oRange.Fields.Add Range:=oRange, Type:=wdFieldPage
  Set oRange = oDoc.Sections(1).Footers(wdHeaderFooterPrimary).Range
  oRange.InsertAfter " von "
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  oRange.Fields.Add Range:=oRange, Type:=wdFieldNumPages
End Sub
```

(Mit der SetRange-Methode manipulieren Sie Start und Ende eines Range-Objektes.)

Arbeiten mit der Eigenschaft FormattedText

▲ Zurück zum Anfang

Wenn Sie Text in einem Word-Dokument transferieren wollen und Wert darauf legen, dass die Formatierung erhalten bleibt, dürfen Sie diesen Text zu keinem Zeitpunkt in einer *String*-Variablen zwischenlagern, weil dadurch die Formatierung unwiderruflich verloren geht.

Sie könnten den Transfer unter zur Hilfenahme der Zwischenablage tätigen, den diese Windows-Komponente ist wesentlich mehr als ein Zwischenspeicher für Zeichenfolgen oder Bildchen. Sie ermöglicht nämlich dem Benutzer das transferieren von Daten in ihrer ursprünglichen Beschaffenheit und dies auf Wunsch meist auch anwendungsübergreifend.

Der programmtechnisch Einsatz der Zwischenablage ist zwar meist ohne grosse Probleme möglich hat aber gewichtige Nachteile, was die folgende Aufzählung verdeutlichen soll.

- Die Zwischenablage gehört eigentlich dem Benutzer. Wird sie ex Code auch gebraucht, verliert der Benutzer die Übersicht und das Vertrauen, was deren Inhalt betrifft.
- Das *ClipBoard*-Objekt existiert zwar in *Visual Basic* nicht aber in VBA. Insbesondere ist dadurch die Prüfung was für Daten (wenn überhaupt) die Zwischenablage enthält nicht möglich (es sei denn Sie benutzen ein *Windows-API*).
- Zumindest was Word 97 betrifft, steht genau eine Zwischenablage zur Verfügung, wo gegen Sie beliebig viele Range-Objekte bilden können.

Sie können einen Bereich in einem Word-Dokument innerhalb des selben Dokumentes oder aber auch zwischen offenen Word-Dokumenten transferieren, indem Sie die Eigenschaft ***FormattedText*** einsetzen. Achten Sie darauf, dass sowohl der Quellbereich, wie auch der Zielbereich mit dieser Eigenschaft arbeitet.

Im folgenden Beispiel wird der erste Absatz im Hauptteil des Dokumentes am Ende dieses Dokumentes dupliziert. Dabei bleibt die Formatierung erhalten.

```
Sub TextTransferieren1()
  Dim oRange As Range
  Set oRange = ActiveDocument.Range
  If Not ActiveDocument.Paragraphs.Last.Range.Text = Chr(13) Then
    oRange.InsertAfter vbCrLf
  End If
  oRange.SetRange Start:=oRange.End + 1, End:=oRange.End + 1
  oRange.FormattedText = ActiveDocument.Paragraphs(1).Range.FormattedText
End Sub
```

(Der erste Absatz im Hauptteil des Dokumentes wird an des Ende des Dokumentes kopiert. Bei Bedarf, wird noch eine Absatzmarke beigefügt.)

Formatierten Text transferieren

▲ Zurück zu den Aufgabenstellungen

Im folgenden Beispiel wird der Hauptteil des aktiven Dokumentes in ein neues, leeres Dokument kopiert, welches auf der selben Dokumentvorlage beruhen soll.

```
Sub FormatierterTextInEinNeuesDokumentTransferieren()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add(Template:=oDoc.AttachedTemplate.FullName)
  nDoc.Content.FormattedText = oDoc.Content.FormattedText
End Sub
```

(Der Hauptteil des aktiven Dokumentes wird in ein neues Dokument kopiert. Das neue Dokument wird mit der selben Dokumentvorlage generiert. Die Formatierung bleibt bei dieser Mimik erhalten.)

Im letzten Beispiel dieses Kapitels wird der gesamte Fussnotentext (also der Text aller Fussnoten) in den Hauptteil eines neuen Dokumentes kopiert. Die Formatierung des Textnotentextes bleibt dabei erhalten. Die Nummerierung der Fussnoten gerät allerdings durcheinander, weil im neuen Dokument die entsprechenden Referenzen fehlen.

```
Sub FussnotentextInNeuesDokument()
  Dim oDoc As Document, nDoc As Document
  Set oDoc = ActiveDocument
  Set nDoc = Documents.Add(Template:=oDoc.AttachedTemplate.FullName)
  nDoc.Content.FormattedText = oDoc.StoryRanges(wdFootnotesStory).FormattedText
End Sub
```

(Der Fussnotentext aller Fussnoten wird ein neues Dokument kopiert. Dabei bleibt die Formatierung des Textes erhalten. Die Nummerierung der Fussnoten gerät allerdings durcheinander, weil die entsprechenden Referenzen im neuen Dokument fehlen.

Arbeiten mit Textmarken

▲ Zurück zum Anfang

Der geschickte Einsatz von Textmarken, kann die Programmierung in MS Word wesentlich einfacher und robuster gestalten. Dazu müssen Sie die Einfügemarke nicht im Dokument bewegen oder sogar code-technisch in eine andere Ansicht (z.B. Kopf- und Fusszeilenansicht wechseln. Sie können die Textmarke nämlich über ihren *Range* ansprechen.

Wegen des hohen Stellenwertes von Textmarken im Zusammenhang mit VBA-Word und weil dieses Thema etwas umfangreich ist, gibt es dazu ein eigenes Skript in dieser Reihe. Informationen zum Arbeiten mit Textmarken erhalten Sie unter dem folgenden Link:

► VBA-technischer Umgang mit Textmarken

Arbeiten mit Zeilen und Seiten

▲ Zurück zum Anfang

Wenn Sie die Tabelle, welche eine Übersicht über die Objekte mit Range-Eigenschaft, aufmerksam studiert haben, dann wird Ihnen vielleicht aufgefallen sein, dass die «Objekte» Zeilen und Seiten gesondert aufgeführt sind.

Das hat seinen Grund, handelt es sich doch bei Zeilen und Seiten im Sinne des Word-Objekt-Modells nicht um Objekte. Text fliesst und bei Fliesstext geschehen Zeilen- und Seitenumbrüche dynamisch, wie von Geisterhand gesteuert. Sie müssen hier auf eine etwas weniger bekannte Methode ausweichen und mit den sogenannten Word-integrierten Textmarken arbeiten.

Dieses Vorgehen hat einen gewichtigen Nachteil, sind diese Textmarken doch nicht eindeutig, sondern wiederholen sich so soft, wie es Zeilen bzw. Seiten im Dokument gibt. Somit muss die Einfügemarke meistens zuerst bewegt werden; nämlich zur fraglichen Zeile bzw. zur fraglichen Seite. Es muss erwähnt werden, dass es dazu keine gute Alternative gibt. Sie sind auf diese Textmarken angewiesen und können Seiten und Zeilen nicht als Objekte ansprechen.

Wenn es sich irgendwie machen lässt, verhindern Sie die Situation überhaupt mit Zeilen und Seiten arbeiten zu müssen. Weichen Sie besser auf Abschnitte (statt Seiten) bzw. Absätze (statt Zeilen) aus. Diese Elemente sind nämlich veritable Objekte im Word-Objekt-Modell und können deshalb einfach, sicher und performand gehandhabt werden.

Zeilen (Line)

▲ Zurück zur Übersicht der Objekte

Im folgenden Beispiel wird der Text der Zeile zurückgegeben, in der sich die Einfügemarke befindet.

```
Sub ZeileZurueckgeben()
  If ActiveDocument.Bookmarks.Exists("\Line") Then
    MsgBox ActiveDocument.Bookmarks("\Line").Range.Text
  End If
End Sub
```

(Der Text der Zeile, in welcher sich die Einfügemarke befindet, wird zurück gegeben.)

Seiten (Page)

▲ Zurück zur Übersicht der Objekte

Eine Seite markieren

▲ Zurück zu den Aufgabenstellungen

Im nächsten Beispiel wird die Seite 3 des aktiven Dokumentes markiert.

```
Sub SeiteNachRange()
  savEnv = Options.Pagination
  Options.Pagination = False
  If ActiveDocument.ComputeStatistics(wdStatisticPages) < 3 Then
    MsgBox "Die betreffende Seite ist nicht vorhanden."
    Exit Sub
  End If
  Selection.GoTo What:=wdGoToPage, which:=wdGoToAbsolute, Count:=3
  Selection.Bookmarks("\Page").Select
  Options.Pagination = savEnv
End Sub
```

(Die dritte Seite im aktiven Dokument wird markiert.)

Das Thema rund um Zeilen und Seiten hat es in sich. Diese Elemente sind einerseits *tricky* in der Handhabung und anderseits *glänzt* MS Word leider auch auch durch Fehlverhalten was den Umgang mit Seiten betrifft. Da das Gebiet umfangreich ist, wird es in einem gesonderten Skript dieser Reihe behandelt. Dieses Skript finden Sie unter dem folgenden Link:

► **Vom Umgang mit Zeilen und Seiten in MS Word**

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► **VBA-technischer Umgang mit Textmarken**
► **VBA-technischer Umgang mit Word-Tabellen**
► **Textgebundene und Textungebundene Grafiken und Objekte ("InlineShapes" und "Shapes")**
► **Vom Umgang mit Zeilen und Seiten in MS Word**
► **Feststellen, wo sich die Einfügemarke befindet**

Microsoft

► **WD2000: StoryRanges Does Not Return Text Box Text in Header/Footer**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Range.htm

René Probst, Februar 2001

Änderungen

- --- nil ---

Vom Umgang mit Zeilen und Seiten in MS Word

◄ Zurück zum Hauptindex

Problem

> 1. Ich möchte eine VBA-Schleife bauen, welche es mir erlaubt, alle Zeilen in einem Word-Dokument abzuarbeiten. Wie stelle ich das an und wie erkenne ich das Ende des Dokumentes?.
> 2. Ich möchte in ein Word-Dokument einen kurzen Text mittels VBA einfügen. Dieser Text soll im 3. Absatz, 2. Zeile, Position 10 erscheinen.
> 3. Ich suche nach einer Routine, welche es mir erlaubt, auf bequeme Art und Weise eine Seite im Dokument vor bzw. hinter eine andere Seite im Dokument zu verschieben.

Diese Seite

- ▼ Einleitung
- ▼ Zeilen- und Seitenumbrüche
- ▼ Vorläufiges Fazit
- ▼ Terminologie
- ▼ Das Kreuz mit Kopf-/und Fusszeilen
- ▼ Dank vordefinierten Textmarken Bereiche (Ranges) bilden
- ▼ VBA-technischer Umgang mit Zeilen
- ▼ VBA-technischer Umgang mit Absätzen
- ▼ VBA-Beispiel mit Fliesstext
- ▼ Warnung: MS Word steht mit Seiten auf dem Kriegsfuss!
- ▼ VBA-technischer Umgang mit Seiten
- ▼ Eine Seite im Dokument löschen
- ▼ Eine Seite des Dokuments in ein neues Dokument übertragen
- ▼ Jede Seite eines Dokuments in einem neuen Dokument abspeichern
- ▼ Tabelle der vordefinierten Textmarken in MS Word
- ▼ Links zu diesem Thema

Einleitung

▲ Zurück zum Anfang

Wenn diese oder ähnliche Fragen aufkommen, dann ist es sehr wohl möglich, dass Sie leider etwas falsch gemacht haben, bzw. dass Sie im Sinne des Word-Objekt-Modells falsch an die Sache herangehen.

Es gibt aber auch Problemstellungen, bei denen man keine gute Alternative hat, sondern gezwungen ist mit den Elementen Zeilen und Seiten zu arbeiten, welche in MS Word nun mal schlecht fassbar sind. Bis zu einem gewissen Masse ist es auch möglich, mit diesen Elementen eines Dokumentes zu jonglieren.

Verhindern Sie aber bereits bei der Planung von komplexen Dokumenten, welche allenfalls systematisch (z.B. mit Code) nachbearbeitet werden sollen, **Zeilen oder Seiten** die Bedeutung von fassbaren Objekten zukommen zu lassen. Dies widerspricht grundsätzlich dem Konzept von MS Word und anderen Textprozessoren.

Denke abstrakt!
Um diese Aussagen verstehen zu können, ist es wohl notwendig sich sowohl mit der Terminologie, sowie auch mit dem Objekt-Model von MS Word auseinander zu setzen. Auch sollte man hinter einem Word-Dokument nicht genau das sehen, was sich auf dem Bildschirm präsentiert, sondern sein Augemerk auf die interne Struktur eines Word-Dokumentes werfen.

Verstehen, was ein Textprozessor ist bzw. macht
Die Hauptnahrung eines Textprozessor ist Fliesstext. Es wird vom Benutzer erwartet, dass dieser fröhlich drauf los schreibt, ohne auf das Ende der Zeile zu achten. Der Textumbruch am Ende der Zeile passiert ja dann automatisch. Dass damit der Text an einer zufälligen Stelle umgebrochen wird, nimmt man auch bei anspruchsvoll gestalteten Texten in Kauf. Oft steht zum Zeitpunkt des Schreibens gar noch nicht fest, ob der Ausdruck auf A4 normierte Briefbögen geschehen soll. Daran erkennen wir, dass eine Zeile für einen Textprozessor keine fassbare Grösse darstellt, wenn auch nicht bestritten wird, dass diese Zeilen optisch klar erkennbar sind.

Sehr ähnlich verhält es sich mit Seitenumbrüchen. Auch hier kann beim Arbeiten mit einem Textprozessor frisch drauf los geschrieben werden. Wieder sorgt Word dafür, dass diese Seitenumbrüche dynamisch geschehen. Dieses Verhalten bewirkt auch hier, dass diese Seitenumbrüche an zufälliger Stelle passieren.

Zeilen- und Seitenumbrüche

▲ Zurück zum Anfang

Unbestreitbar ist, dass es möglich ist, willentlich Zeilenumbrüche zu generieren. Dies kann mit einer manuellen Zeilenumschaltung geschehen (Tastenkombination **[Umschalt + Enter]**). Ganz bestimmt erfolgt auch ein forcierter Umbruch, wenn ein neuer Absatz begonnen wird (Taste [**Enter**]).

Wogegen der manuelle Zeilenumbruch meistens ausschliesslich eine Massnahme zur Textgestaltung ist, ist die Bedeutung eines Absatzwechsel weitreichender. Absätze dienen dazu den Text ordentlich zu gliedern und gönnen dem Leser eine Atempause, um sich auf einen neuen Gedankengang einzustellen.

Ich persönlich verfechte die Meinung, dass manuelle Textumbrüche nur für Spezialfälle sinnvoll sind, und dass ein Schreibstil, welcher sich auf den Grundsatz stützt, **ein Absatz = ein Gedankengang**, nie zu einem Desaster führen kann. **Manuelle Zeilenwechsel innerhalb eines Absatzes, sind in der Regel überflüssig und zu vermeiden**.

Etwas differenzierter sehe ich dies bei Seitenumbrüchen. Hier kommt es sehr auf die Art des Schriftstückes an. In einem flüssig geschriebenen Roman, ist der Stellenwert des Seitenumbruches oft gleich Null, also niedriger als ein Absatzwechsel. Bei einer anderen Kategorie von Schriftstücken, drücken Seitenumbrüche mehr aus, als die Tatsache, dass jedes Stück Papier irgendwann mal voll geschrieben ist. So ist es doch eine gängige und vertraute Praxis, z.B. ein Hauptkapitel auf einer neuen Seite beginnen zu lassen.

Wenn Sie manuelle Seitenumbrüche einsetzen, sollten Sie sich aber im klaren sein, dass die technische Handhabung solch gestalteter Schriftstück damit tendenziell nicht einfacher, sondern komplexer wird.

Wer sich unter einer Seite in MS Word ein ähnliches klar fassbares Element vorstellt, wie es z.B. ein Datensatz in einer Datenbank, oder sich dahinter ein Gebilde wie ein Tabelleblatt (diese Tabellen können übrigens auch Seitenumbrüche enthalten, welche ebenfalls dynamisch und zufällig sind) in einer Excel-Arbeitsmappe erkennt, begibt sich aufs Glatteis. Eine Seite in MS Word ist und bleibt in erster Linie eine pragmatische Massnahme (das Blatt ist voll) in zweiter Linie ist es ein gestalterisches Element. Vom technischen Standpunkt aus betrachtet, ist eine solche Seite kein autonomes Gebilde, welches sich beliebig und kontextunabhängig neu einordnen liesse.

Vorläufiges Fazit

▲ Zurück zum Anfang

- Im Objektmodell von MS Word ist kein Platz für die Einheiten Zeile (ausgenommen Tabellenzeilen) und Seite. MS Word kennt dagegen Zeilen- und Seitenumbrüche. Kein Objekt Zeile aber Zeilenumbrüche, kein Objekt Seite aber Seitenumbrüche: das mag pragmatisch betrachtet spitzfindig sein, im technisches Sinne ist dies allerdings eine ganz grundsätzliche Unterscheidung.
- Weil es gar keine Zeilen und Seiten in MS Word gibt, gestaltet sich der technische Umgang - mit diesen nur optisch wirksamen Elementen - als unpraktisch, fehleranfällig, schwierig und ineffizient.
- Wer hinter Zeilen oder Seiten mehr sieht als gestalterische Elemente, gibt sich auf Glatteis. Wer hinter Zeilen und Seiten sogar logische Strukturelemente vermutet, ist auf dem Holzweg!
- Bezogen auf das Objektmodell von Word, stellen ausschliesslich **Absätze** (statt Zeilenumbrüche) und **Abschnitte** (statt Seitenumbrüche) klar fassbare logische Struktureinheiten dar.

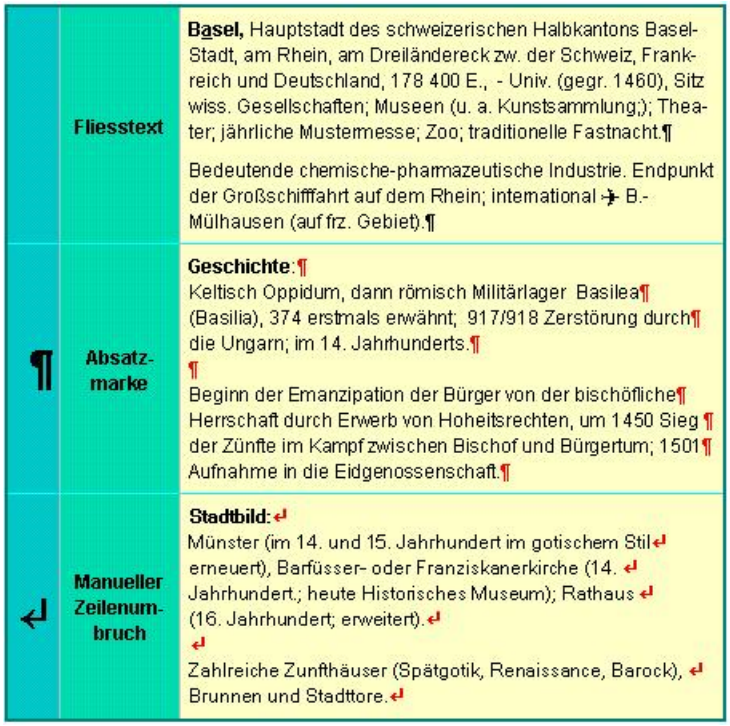
Terminologie

▲ Zurück zum Anfang

Arten von Zeilenumbrüche

MS Word kennt folgende drei Arten von Zeilenumbrüchen:

- Automatische (zufällige) Zeilenumbrüche als Folge von Fliesstext.
- **Manuelle** Zeilenumbrüche (**[Umschalt + Enter]**).
- **Absatzwechsel** (**[Enter]**) impliziert selbstverständlich ebenfalls einen forcierten Zeilenumbruch. In der Regel hat man das Absatzformat so eingerichtet, dass sich ein neuer Absatz optisch mit zusätzlichem Abstand (Durchschuss) vom vorhergehenden Absatz **absetzt**.



Erläuterungen

- Bezogen auf das Objektmodell von Word, stellt ausschliesslich der Absatz eine klar fassbare logische Struktureinheit dar.
- Lädt man Text z.B. in eine Variable, so erkennt man den manuellen Zeilenwechsel als Steuerzeichen dezimal 11 (Hex 0B) und der Absatzwechsel als dezimal 13 (Hex 0D) wieder.
- Ein Absatz ist in MS Word klar definiert. Er beginnt ganz am Anfang des Dokumententeils oder unmittelbar nach einer Absatzmarke und endet mit der nächsten Absatzmarke (inklusive).
- Entfernt man eine Absatzmarke, wird ein Absatz zerstört (also aufgehoben). Dabei geht in der Regel auch ein Stück Absatzformatierung verloren. Dies hat also nicht nur gestalterische Konsequenzen, sondern die Summe der Absätze ist dann nicht mehr gleich. Spricht man eine Sammlung von Absätzen über ihren sequentiellen Index an, muss berücksichtigt werden, dass beim Löschen von Absatzmarken die Absätze weiter hinten nachrücken. Der selbe Effekt tritt natürlich auch dann ein, wenn der Absatz als ganzes entfernt wird, ist die Absatzmarke doch Bestandteil des Absatzes.

Arten von Seitenumbrüche

MS Word kennt folgende Arten von Seitenumbrüchen:

- Automatischer (zufälliger) Seitenumbruch als Folge von Fliesstext.
- Manueller (forcierter) Seitenumbruch (**[Strg + Enter]**).
- Absatzformat (***Format - Absatz - Zeilen- und Seitenwechsel - Seitenwechsel oberhalb***).
- **Abschnittwechsel - Nächste Seite** und deren Derivate namentlich Abschnittwechsel - Ungerade/Gerade Seite. Dies Seitenwechsel werden über ***Einfügen - Manueller Wechsel*** in das Dokument eingebracht.

Erläuterungen

- Ausschliesslich Abschnitte stellen Elemente des Word-Objektmodells dar, nicht aber Seiten.
- Die konsistente Zuordnung von Kopf-/Fusszeilen ist nur zu Abschnitten möglich, nicht aber zu Seiten. Dabei ist zu bedenken, dass jeder Abschnitt über bis zu drei Kopfzeilenbereiche verfügen kann, dann nämlich wenn die Option Erste Seite anders und/oder die Option Gerade/Ungerade Seiten anders eingeschalten ist. Zudem ist zu bedenken, dass die Kopf-/Fusszeilen von sich folgenden Abschnitten verbunden sein können
- Manuelle Seitenwechsel sind zu vermeiden. Verwenden Sie an deren Stelle besser das ***Absatzformat Seitenwechsel oberhalb***. Auch Word2000 macht immer noch Fehler im Umgang mit manuellen Seitenwechseln (z.B. bei der automatischen Generierung von Inhaltsverzeichnissen unter bestimmten Konditionen). Microsoft kriegt das nicht richtig in den Griff, obwohl es sich dabei um eine Grundfunktionen der Textverarbeitung handelt. Eher kann MS Word quadratische Kreise generieren, als in allen Fällen korrekt mit Seiten um zu gehen. Auch Sie werden sich vermutlich die Zähne ausbeissen, wenn Sie komplexere Routinen auf die Beine stellen möchten, welche in diesem Bereich aktiv werden. Siehe dazu auch das Kapitel Die wirklichen Schnitzer von Microsoft in diesem Skript.

Das Kreuz mit Kopf-/und Fusszeilen

▲ Zurück zum Anfang

Wie hier schon erwähnt wurde, ist es nicht in allen Fällen möglich, Kopf- und Fusszeilen eindeutig einer Seite zuzuordnen. Diese Zuordnung kann gar nicht in allen Fällen eindeutig geschehen. Was soll z.B. geschehen, wenn eine Seite inklusive Kopfzeile in ein anderes Dokument einkopiert wird, welches ebenfalls über eine Kopfzeile verfügt?

Die Antwort auf diese Frage lautet, es kommt auf die Situation an, oder anders ausgedrückt, der Mensch müsste hier entscheiden können. Das bedingt selbstverständlich, dass er sich erst mal dieser Problematik bewusst ist.

Eigentlich wäre genau dies eine Situation, in welcher sich der Office-Assistent profilieren könnte und auf Grund einer Analyse nachfragen, welche möglichen und halbwegs sinnvollen Szenarien jetzt zur Verfügung stehen.

Aber Pusteblume, der Office-Assistent langweilt uns mit solch grundlegenden Frage wie: *Wollen Sie einen Brief schreiben?* und ähnlichem. Er ist mehr der Pausenclown als eine echte Hilfe in kritischen Situationen. Folglich sind Sie gefordert. Bedenken Sie dabei folgendes:

- **Kopf-/und Fusszeilenbereiche** sind nicht Elemente von Seiten, sie sind **Elemente von Abschnitten**. Sie können gar keine Elemente von Seiten sein, weil eine Seite gar nichts fassbares im Objektmodell von Word darstellt.
- Es kommt noch spannender: Kopf-/und Fusszeilenbereiche liegen gar nicht auf der selben Ebene wie der übrige Teil des Dokumentes. Sie stellen eine eigene Ebene dar, in der Word-Fachsprache werden diese Ebenen **StoryRanges** genannt.
- Jeder Abschnitt kann bis zu drei Kopfzeilenbereiche und bis zu drei Fusszeilenbereiche aufweisen, namentlich: **Primäre Kopf-/Fusszeile**, **Erste Seite, Kopf-/Fusszeile** und **Kopf-/Fusszeile für gerade Seiten**.
- Es ist leider nicht immer klar, wo eine Seite beginnt bzw. wo diese aufhört. Gehört nun der Seitenumbruch selbst zur Seite, welche avisiert wird oder gehört dieser bereits zur folgenden Seite? Denkbar ist auch ein weder noch, also der Seitenumbruch selbst ist eigenständig. Diese und ähnliche Fragen haben mich in der Vergangenheit oft beschäftig. Es scheint sich um ein zwitteriges Wesen zu handeln; wie will nun dies Spezi angesprochen werden?

Dank vordefinierten Textmarken Bereiche (*Ranges*) bilden

▲ Zurück zum Anfang

Der VBA-technische Umgang mit Zeilen und Seiten ist also oft ein steiniger Weg. Trotzdem wollen wie nun ran an den Speck. Da wir nicht mit Objekten arbeiten können, müssen wir auf eine andere Option ausweichen, welche uns Word-VBA bietet. Dabei machen wir uns zu Nutze, dass es in einem Word-Dokument vordefinierte und versteckte Textmarkenfelder gibt. Dank ihnen lässt sich der Inhalt von Zeilen und Seiten ansteuern. Eine komplette Liste dieser vordefinierten Textmarken finden Sie übrigens etwas weiter unten in diesem Skript. Beachten Sie, dass es sich dabei nicht um einen sehr performanden Weg handelt. Die ist ein Grund mehr, sich sehr früh Gedanken zu machen, wie man dieser Situation ausweichen kann. Als Alternative bieten sich übrigens Absätze statt Zeilen bzw. Abschnitte statt Seiten an.

Vordefinierte Textmarken

Üblicherweise sind Textmarkennamen eindeutig in einem Dokument. Das will heissen, es gibt nur eine Textmarke mit dem selben Namen. Bei den vordefinierten Textmarken ist das aber oft anders. Es gibt für jede Zeile eine Textmarke mit dem selben Namen. Das gleiche gilt für Textmarkenfelder, welche für Seiten stehen.

Ob diese Textmarken im Format des Dokumentes wirklich vorhanden sind oder ob es sich nun um eine symbolische Angelegenheit handelt, lassen wir mal dahingestellt. Für uns ist wichtig, dass der Zugriff auf Zeilen oder Seiten über diese Textmarken tatsächlich gelingt.

Da es aber viele gleichnamige Textmarken dieser Art im Dokument gibt, müssen wir zuerst zur richtigen Textmarke navigieren. Das wiederum will heissen, wenn wir z.B. den Inhalt der dritten Seite des Dokumentes zurück bekommen wollen, müssen wir zur dritten Seite hüpfen und dann den Inhalt der Seite ansprechen.

Fortgeschrittene VBA-Programmierer wissen, dass das Navigieren innerhalb eines Dokumentes eigentlich verpönt ist. Tatsächlich ist dies auch problematisch, denn erstes ist es eine langsame Methode und zweitens ist die Methode nicht immer anwendbar. Insbesondere, wenn Word von einer anderen Anwendung ferngesteuert wird, ist das Navigieren in einem Word-Dokument nicht das Gelbe vom Ei. Allerdings haben wir keine direkte Alternative, wenn es darum geht, Zeilen oder Seiten anzusteuern. Vor dieser Situation stehen wir immer dann, wenn wir keine bzw. die falschen Vorkehrungen gemacht habe, oder wenn wir trotz sorgfältiger Planung in die Defensive gelangen - uns also die guten Alternativen verbaut haben.

Bilden von Range-Objekten

Immerhin, sobald wir am richtigen Ort sind, können wir einen *Range* bilden und somit dann mit einem Bereich arbeiten. Das arbeiten mit Bereichen ist nämlich wesentlich effizienter, als das dauernde herum hüpfen in einem Word-Dokument.

VBA-technischer Umgang mit Zeilen

▲ Zurück zum Anfang

Das folgende Beispiel markiert die Zeile, in dem sich die Einfügemarke gerade befindet:

```
Sub AktuelleZeileMarkieren1()
  ActiveDocument.Bookmarks("\Line").Select
End Sub
```

(Markiert die Zeile in welcher sich die Einfügemarke befindet.)

Da diese Methode fehl schlagen könnte, z.B. wenn die Einfügemarke sich in einem Textfeld befindet gehen wir etwas kritischer und sorgfältiger an die Sache heran:

```
Sub AktuelleZeileMarkieren2()
  If ActiveDocument.Bookmarks.Exists("\Line") Then
    ActiveDocument.Bookmarks("\Line").Select
  End If
End Sub
```

(Markiert die Zeile in welcher sich die Einfügemarke befindet. Vorsichtshalber wird eine Abfrage vorgeschaltet.)

Folgendes Beispiel ist eine allgemeingültige Routine. Sie können den Index des Absatzes, sowie die Zeile innerhalb dieses Absatzes und schliesslich noch die Position innerhalb der Zeile vorgeben. An dieser Stelle wird dann die Zeichenfolge "Bingo!" eingefügt.

```
Sub AbsatzZeilePosition()
  Absatz = 5
  Zeile = 4
  Position = 6
  Dim oRange As Range
  If Zeile = 1 Then
    Set oRange = ActiveDocument.Paragraphs(Absatz).Range
  Else
    Set oRange = _
       ActiveDocument.Paragraphs(Absatz).Range.GoTo(What:=wdGoToLine, _
          Which:=wdGoToRelative, Count:=Zeile - 1)
  End If
  oRange.Start = oRange.Start + Position - 1
  oRange.InsertAfter "Bingo!"
End Sub
```

(Diese Routine zielt auf die vierte Zeile im fünften Absatz des Dokumentes. In dieser Zeile an Position sechs wird eine Zeichenfolge eingefügt.)

Sollte der fünfte Absatz über weniger als vier Zeile verfügen. Dann geht die Rechnung nicht mehr richtig auf. Nach Möglichkeit sollten Sie auf solche Übungen verzichten. In diesem konkreten Beispiel würde ich das Arbeiten mit einer eigens definierten Textmarke, welches diese Position markiert, klar vorziehen.

Beim Arbeiten mit Textmarken bzw. Textmarkenfeldern, erübrigt sich nämlich auch die lästige Positionierung. Wenn immer Möglich, sollten Sie eine Lösung anstreben, bei der die Einfügemarke nicht bewegt werden muss, weder durch eine *GoTo-Anweisung* noch mittels *MoveDown*- und ähnlichen Methoden.

Oft wird angefragt, wie man ein Word-Dokument vom Anfang bis zum Ende zeilenweise auslesen kann. Auch hier gilt als einleitende Bemerkung, dass wenn es sich bei diesen Zeile um Absätze handeln sollte, dann sollten Sie unbedingt Absätze und nicht Zeilen einlesen. Methoden um Absätze einzulesen, finden Sie im anschliessenden Kapitel.

Sollten Sie nun aber ein Word-Dokument geerbt haben, welches mit manuellen Zeilenumschaltungen formatiert ist, dann können Sie diese Zeilen wie folgt sequentiell abarbeiten:

```
Sub ZeilenSequentielAbarbeiten()
  ActiveDocument.Range(0, 0).Select 'An den Anfang positionieren
  While Not Selection.Bookmarks.Exists("\EndOfDoc") 'Bis zum Ende des Dokumentes
    MsgBox Selection.Bookmarks("\Line").Range.Text 'Zeile zurückgeben
    Selection.MoveDown Unit:=wdLine 'Zur nächsten Zeile positionieren
  Wend
End Sub
```

(Sequentielles Abarbeiten von Zeilen in einem Dokument.)

Im folgenden wird ein Absatz, welcher Zeilenumbrüche enthält eingelesen. Der Absatz wird dann aufgeteilt, so dass für jede Zeile eine Variable gebildet wird. Da die **Split-Anweisung** erst mit **Word2000** unterstützt wird. wird das folgende Makro in dieser Form bei **Word 97** nicht zum Erfolg führen.

```
Sub AbsatzInZeilenSpalten()
  Dim Zeilen() As String
  Zeilen = Split(ActiveDocument.Paragraphs(1).Range.Text, Chr(11))
  For i = 0 To UBound(Zeilen)
    MsgBox Zeilen(i)
  Next i
End Sub
```

(Der erste Absatz des Dokumentes, welcher manuelle Zeilenumbrüche enthält, wird in seine Zeilen aufgespaltet. Diese Methode funktioniert erst ab Word2000.)

Sie können diese Methode auch auf ein ganzes Word-Dokument anwenden. Der Code um den ganzen Hauptteil des Word-Dokumentes, welcher manuelle Zeilenumbrüche aufwiest in seine Zeilen aufzuspalten sieht wie folgt aus:

```
Sub DokumentInZeilenSpalten1()
  Dim Zeilen() As String
  Zeilen = Split(ActiveDocument.Content, Chr(11))
  For i = 0 To UBound(Zeilen)
    MsgBox Zeilen(i)
  Next i
End Sub
```

(Der ganze Hauptteil des Dokumentes, welcher manuelle Zeilenumbrüche enthält, wird in seine Zeilen aufgespaltet. Diese Methode funktioniert erst ab Word2000.)

Angenommen es liegt Ihnen ein Dokument vor, welches in gemischter Weise manuelle Zeilenumbrüche und Absatzmarken beinhaltet, dann könnten Sie den Text wie folgt in Zeilen aufspalten:

```
Sub AbsatzInZeileSpalten2()
  Dim Zeilen() As String
  strText = Replace(ActiveDocument.Content, Chr(13), Chr(11))
  Zeilen = Split(strText, Chr(11))
  For i = 0 To UBound(Zeilen)
    MsgBox Zeilen(i)
  Next i
End Sub
```

(Der ganze Hauptteil des Dokumentes, welcher in gemischter Weise manuelle Zeilenumbrüche und Absatzmarken enthält, wird in seine Zeilen aufgespaltet. Diese Methode funktioniert erst ab Word2000.)

VBA-technischer Umgang mit Absätzen

▲ Zurück zum Anfang

Wesentlich erfreulicher ist das Arbeiten mit Absätzen, stellen diese doch veritable Objekte des Word-Objekt-Modells dar.

Beim sequentiellen einlesen von Absätzen bieten sich Ihnen sogar mehrere Möglichkeiten, welche alle performand und unproblematisch sind, da eine Positionierung bzw. eine Navigation entfällt. Dies will heissen, dass Sie z.B. den Absatz Nummer 23 auch dann ansprechen können, wenn sich die Einfügemarke an einem ganz anderen Ort im Dokument befindet. Auch muss die Einfügemarke nicht bewegt werden, um zu einem anderen Absatz zu gelangen.

Im folgenden wird der erste Absatz (Absatzmarke inklusive) des Dokumentes in eine Variable geladen:

```
Sub ErsterAbsatzInVariable()
  strText = ActiveDocument.Paragraphs.First.Range.Text
End Sub
```

(Der erste Absatz im Dokument wird in eine Variable geladen.)

Im folgenden wird ein Range gebildet, welche den dritten Absatz (exklusive Absatzmarke) beinhaltet:

```
Sub AbsatzNachRangeOhneAbsatzMarke()
  Dim oDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  Set oRange = oDoc.Range(Start:=oDoc.Paragraphs(3).Range.Start, _
     End:=oDoc.Paragraphs(3).Range.End - 1)
  oRange.Select
End Sub
```

(Bildung eines *Ranges*, welcher den dritten Absatz ohne Absatzmarke umfasst.)

Im folgenden wird ein Range gebildet, welcher die Absätze 3-6 des Dokumentes beinhaltet:

```
Sub MehrereAbsätzeNachRange()
  Dim oDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  Set oRange = oDoc.Range(Start:=oDoc.Paragraphs(3).Range.Start, _
     End:=oDoc.Paragraphs(6).Range.End)
  oRange.Select
End Sub
```

(Aus den Absätzen 3-6 des Dokumentes wird ein *Range* gebildet.)

Im folgenden wird die sehr nahe liegende Methode angezeigt, um alle Absätze im Hauptteil des Dokumentes sequentiell einzulesen. Diese Methode sollte dann **nicht** verwendet werden, wenn der Inhalt der Absätze in der Schleife verändert wird. In diesem Fall besteht nämlich die Gefahr, dass mit jeder Änderung die Kollektion neu durchlaufen wird, was in einer sehr unliebsamen Endlosschleife endet.

```
Sub AbsatzeSequentiellEinlesen1()
  Dim Absatz As Paragraph
  For Each Absatz In ActiveDocument.Paragraphs
    MsgBox Absatz.Range.Text
  Next
End Sub
```

(Die Absätze im Dokument werden sequentiell eingelesen. Verwenden Sie diese Methode nicht, falls Sie die Absätze in der Schleife verändern wollen.)

Hier eine Alternative, welche es erlaubt, die Absätze zu verändern. Die Veränderung sollte allerdings die Gesamtzahl der Absätze nicht verändern, sonst ist mit einem Fehler bzw. einem Fehlverhalten zu rechen.

```
Sub AbsatzeSequentiellEinlesen2()
  Dim Absatz As Paragraph
  Anz = ActiveDocument.Paragraphs.Count
  For i = 1 To Anz
    Set Absatz = ActiveDocument.Paragraphs(i)
    Absatz.Range.Text = i & ". " & Absatz.Range.Text
  Next i
End Sub
```

(Diese Schleife, welche die Absätze im Dokument sequentiell abarbeitet bietet mehr Sicherheit, als das erste Beispiel. Trotzdem sollten Sie darauf achten, dass sich die Gesamtzahl der Absätze durch die Zuweisung nicht verändert.)

Die sicherste und eleganteste Lösung ist aber folgende. Dabei darf sich die Anzahl der Absätze während der Verarbeitung auch ändern. Allenfalls können Sie mit der Methode ***Next*** bzw. ***Previous*** nachjustieren. "Logische" Fehler, welche Sie durch intensives Testen ausmerzen müssen, sind zwar möglich. Zu einem unkontrolliertem Abbruch oder zu einer Endlosschleife sollte es allerdings mit dieser robusten Mimik nicht kommen.

```
Sub AbsätzeSequentiellAbarbeiten()
  Dim Absatz As Paragraph
  Set Absatz = ActiveDocument.Paragraphs(1)
  While Not Absatz Is Nothing
    MsgBox Absatz.Range.Text
    Set Absatz = Absatz.Next
  Wend
End Sub
```

(Die letzte und vielleicht eleganteste Lösung um die Absätze in einem Dokument sequentiell abzuarbeiten.)

Zum Schluss dieses Kapitels noch eine Lösung aus der Praxis. Aus einer Anhäufung von Absätzen sollen Doubletten entfernt werden.

Damit sicher gestellt ist, dass allfällige Doubletten genau hintereinander stehen, werden die Absätze im Dokument zuerst alphabetisch sortiert. Anschliessend werden diese Doubletten auf effiziente Art und Weise entfernt.

Das Resultat ist eine sortierte Liste von Absätzen, wobei kein Absatz doppelt vorhanden ist.

```
Sub DoublettenEntfernen()
  ActiveDocument.Content.Sort FieldNumber:="Absätze"
  Dim Absatz As Paragraph
  Set Absatz = ActiveDocument.Paragraphs(1)
  While Not Absatz Is Nothing
    If Absatz.Range.Text = Absatz.Next.Range.Text Then
      Absatz.Next.Range.Delete
      Set Absatz = Absatz.Previous
    End If
    Set Absatz = Absatz.Next
  Wend
End Sub
```

(Entfernt Doubletten aus einer Anhäufung von Absätzen.)

VBA-Beispiel mit Fliesstext

▲ Zurück zum Anfang

Das folgende Beispiel verarbeitet einen Fliesstext. Konkret geht es darum, in einem Dokument bedingte Trennstriche (ANSI Code 31) zu löschen. Allerdings sollen diese Sonderzeichen nur dann gelöscht werden, wenn das Zeichen **nicht** am Ende einer Zeile im Fliesstext steht.

```
Sub BedingtesLoeschenVonTrennstrichen()
  ActiveDocument.Range(0, 0).Select
  With Selection.Find
    .ClearFormatting
    .Text = Chr(31)
    .Execute
    While .Found = True
      x = Selection.Bookmarks("\Line").Range.End
      If Not Selection.Range.End = x Then
        Selection.Font.ColorIndex = wdYellow
        'Selection.Range.Delete
      End If
      .Execute
    Wend
  End With
End Sub
```

(Entfernt sämtliche, nicht endständige Trennstriche innerhalb eines Fliesstextes.)

Warnung: MS Word steht mit Seiten auf dem Kriegsfuss!

▲ Zurück zum Anfang

Der Umgang mit Seiten in einem Word-Dokument ist problematisch. Sollten Sie sogar das Pech haben, einen Auftrag zu fassen, welcher das Ziel verfolgt, einen Vorgang bei dem Seiten im Vordergrund stehen, zu automatisieren, dann kann das sehr schnell zum Alptraum werden. Dies trifft insbesondere dann zu, wenn die Routine unabhängig vom vorliegenden Dokument zu konsistenten und reproduzierbaren Ergebnissen führen soll, also den Anspruch auf Allgemeingültigkeit stellt.

Die drei neuralgischen Punkte im Umgang mit Seiten

Ich lokalisiere dabei gleich drei neuralgische Punkte, wobei ich sowohl den ersten, wie auch den zweiten Punkt akzeptiere. Was allerdings den dritten Punkt angeht, werfe ich Microsoft vor, ein (wissentlich) fehlerhaftes und damit schlechtes Produkt für gutes Geld zu verhökern.

Asynchronität oder die Tücken einer WYSIWYG-Anwendung

Der erste neuralgische Punkt ist die Asynchronität eines Vorganges, welcher Seitenumbruch genannt wird.

Damit spreche ich die Tatsache an, dass in der Regel z.B. beim Schreiben und auch beim Formatieren, das Dokument immer neu umgebrochen wird. Da der Erfolg von MS Word in einem wesentlichen Teil darin liegt, dass es sich dabei um eine WYSIWYG-Anwendung (What You See Is What You Get) handelt, ist es folgerichtig, dass der Benutzer mit der Irritation, welche von asynchronen Vorgängen ausgehen kann, leben muss.

Materielle und virtuelle Welt

Die zweite Problematik wurde in diesem Skript schon angesprochen. Das Objekt-Modell von Word kennt weder Zeilen noch Seiten. Für uns, welche noch alle mit gedruckten Erzeugnissen aufwuchsen, mag das irritierend sein. Allerdings, wieso soll eine weisse Fläche, welche irgend jemand mal sehr zufällig auf 297mm x 210mm festgelegt hat, auch in einer virtuellen Welt eine fassbare Grösse (im doppelten Sinnes des Wortes) darstellen. Von dieser Vorstellung muss man sich tatsächlich lösen, wenn man technologisch vorwärts schreiten möchte.

MS Word kennt keine Zeilen und keine Seiten. MS Word kennt dagegen Zeilenumbrüche und Seitenumbrüche. Für einen Pragmatiker mag diese Unterscheidung spitzfindig erscheinen, für Analytiker (und Programmierung ist analytisch) ist der Unterschied ganz grundlegend. Die Konsequenzen daraus sind Janus-artig. Es setzt Grenzen, eröffnet aber auch neue Horizonte, wobei das Gewinnpotential höher einzustufen sind als die Verlustmarge.

Die Brücken zwischen materieller und virtueller Welt

Das jonglieren mit Seiten, ist und bleibt schwierig. Mit den vordefinierten Textmarken, auf welche in diesem Skript ausführlich eingegangen wird, hat Microsoft eine Brücke gebaut. Dies ist zu anerkennen und löblich. Trotzdem, man sollte diese Brücke nur dann begehen, wenn keine Alternativen vorhanden sind.

Nicht alles was matt erscheint ist Blei

Nicht immer, wenn sich der Benutzer über eine Eigenart in MS Word aufregt, liegt tatsächlich ein Fehlverhalten vor. Manchmal braucht es wirklich sehr profunde Kenntnisse (darunter auch technischer Natur), um eine solche Eigenart richtig einordnen zu können. Fairerweise müsste man sich ja auch überlegen, was würde passieren, wenn Microsoft die Sache anders in die Hand genommen hätte. Gibt es wirklich ein Konzept, welches möglichst anwendungsweit durchgezogen werden kann und dann unter dem Strich zu konsistenteren Resultate führen würde.

Spielen Sie diesen Gedankengang an einer konkreten Eigenart durch, z.B. an der Tatsache, dass Kopf-/und Fusszeilen nicht direkt an Seiten gebunden sind. Wahrscheinlich kommen Sie zum Schluss, dass es nicht so verkehrt sein kann, was Microsoft hier vorexerziert. Dass dieses Vorgehen auch mit Nachteilen verbunden ist, sei hier nicht bestritten.

Die wirklichen Schnitzer von Microsoft

Mir sind zahlreiche dicke Probleme und Unzulänglichkeiten bekannt, welche sich im Problemkreis MS Word und Seiten bewegen. Diese Fehlverhalten kann ich nicht mit einem hochwertigen Produkt vereinbaren. Einige diese Fehler manifestieren sich ausschliesslich diffus (was den Umgang damit keinen Falls erleichtert). Einige kommen erst so richtig zum Ausdruck, wenn man (intensiv) mit VBA an ein Word-Dokument heran geht. Es existieren aber auch Fehler in MS Word, welche sich ohne grossen Aufwand und ohne VBA immer wieder auf einfache Art und Weise reproduzieren lassen. Von der letzteren Sorte von Fehlern sei hier einer dokumentiert, damit der Leser sich vorstellen kann, wovon ich spreche. Gehen Sie bei der Nachstellung des Fehlers wie folgt vor:

1. Erstellen Sie ein neues Word-Dokument.
2. Schreiben Sie frisch darauf los und füge Sie auch **4 - 5 manuelle Seitenumbrüche** in das Dokument ein.
3. Fügen Sie nun an beliebiger Stelle einen ***Manuellen Abschnittwechsel - Gerade Seite*** ein.
4. Löschen Sie den Abschnittwechsel unverzüglich wieder. (Damit scheint das Problem vorgespurt zu sein).
5. Rufen Sie den Befehl ***Bearbeiten - Gehe Zu - Seite 3*** auf. Word positioniert auf den Anfang der Seite 2.

Sie können nun mit dem Dokument machen und unterlassen, was immer Sie wollen (z.B. Schliessen und wieder Öffnen) Word wird in diesem Dokument von nun an immer falsch positionieren.

Achten Sie nun bitte auch auf die Statusleiste. Das steht nun nämlich links keck Seite 3 (Sie befinden sich allerdings auf Seite 2) und etwas weiter rechts steht dann 2 von 5 was eigentlich korrekt ist, nur Sie wollten ja eigentlich zur Seite 3.

Nach Microsoft liegt das Problem übrigens darin, dass der vorhergehende Abschnitt die Formatierung des folgenden annimmt.

Auch die sechste Generation eines Textprozessors à la Microsoft (Word2000) macht also immer noch Fehler im Umgang mit manuellen Seitenwechseln (z.B. bei der automatischen Generierung von Inhaltsverzeichnissen unter bestimmten Konditionen). Microsoft kriegt diese Angelegenheit offensichtlich nicht richtig in den Griff, obwohl es sich dabei um eine absolute Grundfunktionen der Textverarbeitung handelt. Eher kann MS Word quadratische Kreise generieren, als in allen Fällen korrekt mit Seiten umgehen.

VBA-technischer Umgang mit Seiten

▲ Zurück zum Anfang

Auch der Umgang mit Seiten in MS Word ist ziemlich problematisch, da diese Elemente, wie schon mehrfach hier erwähnt, keine Objekte des Word-Objekt-Modells darstellen. Sie sind auch hier auf vordefinierte Textmarken angewiesen. Aber es kommt noch schlimmer.

Wenn Sie einige Erfahrung mit MS Word gesammelt haben, dann ist Ihnen wahrscheinlich aufgefallen, dass diese Anwendung mit Seitenzahlen ziemlich auf dem Kriegsfuss steht. Dass hat einerseits mit dem *WYSIWYG*-Konzept von MS Word zu tun und anderseits mit der Tatsache, dass der Seitenumbruch in vielen Fällen asynchron geschieht, also zeitgleich mit anderen Vorgängen geschieht (im Gegensatz zu hinter einander).

Wenn die Seitenzahlen in einem Word-Dokument wieder einmal nicht stimmen, heisst es von Seiten Microsoft meist lapidar: *Schalten Sie in die Seitenansicht und wieder zurück. Wiederholen Sie allenfalls diese Aktion allenfalls mehrmals.*

Wogegen dies in einem nicht automatisierten Vorgang noch einigermassen zumutbar ist, ist dies für die Programmierung äusserst gefährlich und fragwürdig, nicht zuletzt deshalb, weil ein Programm Befehle um ein vielfaches schneller ausführt als ein Mensch.

Die bereits erwähnten vordefinierten Textmarken sind nicht eindeutig. Viel mehr erscheinen diese (mit dem selben Namen) so oft in Ihrem Dokument, wie das Dokument Seiten enthält. Dies wiederum hat zur Folge, dass Sie zuerst positionieren müssen, also die Eingabemarke z.B. mit dem *Selection*-Objekt verschieben müssen, eine Mimik also, welche eigentlich in der VBA-Word-Programmierung verpönt ist.

Der folgende Code positioniert die Einfügemarke an den Anfang der dritten Seite des Dokuments. Da sicher gestellt werden muss, dass das Dokument bereits umgebrochen ist, werden zwei Schritte vorgeschaltet. Zum einen wird die Option Seitenumbruch im Hintergrund für die Laufzeit des Prozesses ausgeschalten und anschliessend wieder auf den ursprünglichen Wert zurück gesetzt. Zum zweiten wird das Dokument dann vom Code forciert umgebrochen. Nur so sind korrekte Ergebnisse garantiert. Ein anderes Vorgehen, birgt die Gefahr, dass diese Prozesse asynchron ablaufen und so zu zufälligen Resultaten führen.

```
Sub AufSeitePositionieren()
  savEnv = Options.Pagination
  Options.Pagination = False
  ActiveDocument.Repaginate
  Selection.GoTo What:=wdGoToPage, which:=wdGoToAbsolute, Count:=3
  Options.Pagination = savEnv
End Sub
```

(Positioniert die Einfügemarke am Anfang der dritten Seite des Dokuments.)

Im folgenden Beispiel wird ein *Range* (Bereich) gebildet, welcher den Inhalt der dritten Seite beinhaltet:

```
Sub SeiteNachRange()
  Dim oRange As Range
  savEnv = Options.Pagination
  Options.Pagination = False
  ActiveDocument.Repaginate
  Selection.GoTo What:=wdGoToPage, which:=wdGoToAbsolute, Count:=3
  Set oRange = Selection.Bookmarks("\Page").Range
  Options.Pagination = savEnv
  oRange.Select
End Sub
```

(Eine *Range* wird gebildet, welche die dritte Seite im Dokument beinhaltet.)

Im nächsten Beispiel wird die dritte Seite des Dokuments markiert::

```
Sub SeiteMarkieren()
  savEnv = Options.Pagination
  Options.Pagination = False
  ActiveDocument.Repaginate
  Selection.GoTo What:=wdGoToPage, which:=wdGoToAbsolute, Count:=3
  Selection.Bookmarks("\Page").Range.Select
  Options.Pagination = savEnv
End Sub
```

(Die dritte Seite im Dokument wird markiert.)

Im folgenden Beispiel wird ermittelt, wie viele **Formularfelder** die zweite Seite des Dokumentes aufweist:

```
Sub AnzahlDerFormularfelderEinerSeiteErmitteln()
  Seite = 2
  savEnv = Options.Pagination
  Options.Pagination = False
  ActiveDocument.Repaginate
  Selection.GoTo What:=wdGoToPage, which:=wdGoToNext, Name:=Seite
  MsgBox Selection.Bookmarks("\Page").Range.FormFields.Count
  Options.Pagination = savEnv
  Application.GoBack
End Sub
```

(Ermittelt die Anzahl der Formularfelder auf einer vorgegebenen Seite des Dokumentes)

Eine Seite im Dokument löschen

▲ Zurück zum Anfang

Die folgende Routine soll es Ihnen erleichtern eine von Ihnen vorgegeben Seite im Dokument zu löschen. Sie werden gefragt, welche Seite gelöscht werden soll.:

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Titel = "Seite löschen"

Sub SeiteLoeschen()
  savEnv = Options.Pagination
  Options.Pagination = False
  max = ActiveDocument.ComputeStatistics(wdStatisticPages)
  Flag = False
  While Flag = False
    Text = "Welche Seite soll gelöscht werden?" & vbCrLf
    Text = Text & "1 - " & max
    Antwort = InputBox(Text, Titel, "1")
    If Antwort = "" Then Exit Sub
    Dim A As Integer
    On Error Resume Next
    A = Antwort
    Fehler = Err.Number
    On Error GoTo 0
    If Fehler = 13 Then
      MsgBox "Das ist keine gültige Zahl.", vbInformation, Titel
    Else
      If A > max Or A < 1 Then
        MsgBox "Die Zahl liegt nicht im gültigen Bereich.", vbInformation, Titel
      Else
        Flag = True
      End If
    End If
  Wend
  Selection.GoTo What:=wdGoToPage, which:=wdGoToAbsolute, Count:=Int(A)
  Selection.Bookmarks("\Page").Range.Delete
  Options.Pagination = savEnv
End Sub
```

(Diese Routine erleichtert Ihnen das Löschen einer von Ihnen vorgegeben Seite im Dokument.)

Eine Seite des Dokuments in ein neues Dokument übertragen

▲ Zurück zum Anfang

Die folgende Routine überträgt eine von Ihnen bestimmte Seite im Dokument in ein neues Dokument. Dabei wird die Zwischenablage bewusst **nicht** verwendet. Die Formatierung bleibt dabei erhalten.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Titel = "Seite kopieren"

Sub SeiteKopieren()
  savEnv = Options.Pagination
  Options.Pagination = False
  max = ActiveDocument.ComputeStatistics(wdStatisticPages)
  Flag = False
  While Flag = False
    Text = "Welche Seite soll kopiert werden?" & vbCrLf
    Text = Text & "1 - " & max
    Antwort = InputBox(Text, Titel, "1")
    If Antwort = "" Then Exit Sub
    Dim A As Integer
    On Error Resume Next
    A = Antwort
    Fehler = Err.Number
    On Error GoTo 0
    If Fehler = 13 Then
      MsgBox "Das ist keine gültige Zahl.", vbInformation, Titel
    Else
      If A > max Or A < 1 Then
        MsgBox "Die Zahl liegt nicht im gültigen Bereich.", vbInformation, Titel
      Else
        Flag = True
      End If
    End If
  Wend
  Selection.GoTo What:=wdGoToPage, which:=wdGoToAbsolute, Count:=Int(A)
  Dim nDoc As Document, oRange As Range
  Set oRange = Selection.Bookmarks("\Page").Range
  If Right(oRange.Text, 1) = Chr(12) Then
    oRange.SetRange Start:=oRange.Start, End:=oRange.End - 1
  End If
  Set nDoc = Documents.Add(Template:=ActiveDocument.AttachedTemplate.FullName)
  nDoc.Content.FormattedText = oRange.FormattedText
End Sub
```

(Diese Routine erleichtert Ihnen das Kopieren einer von Ihnen vorgegeben Seite im Dokument. Dabei bleibt die Zwischenablage unbehelligt.)

Jede Seite eines Dokuments in einem neuen Dokument abspeichern

▲ Zurück zum Anfang

Der folgende VBA-Code speichert jede einzelne Seite des aktiven Dokumentes in ein neues Dokument ab. Dabei erhalten die neuen Dokumente einen Dateinamen, welcher sich zusammensetzt aus dem Präfix, welchen Sie im Kopf des Codes festlegen und einer fortlaufenden Nummer.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Präfix = "D"

Sub JedeSeiteEinNeuesDokument()
  savEnv = Options.Pagination
  Options.Pagination = False
  max = ActiveDocument.ComputeStatistics(wdStatisticPages)
  Dim oDoc As Document, nDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  For i = 1 To max
    oDoc.Activate
    Selection.GoTo What:=wdGoToPage, which:=wdGoToAbsolute, Count:=i
    Set oRange = Selection.Bookmarks("\Page").Range
    If Right(oRange.Text, 1) = Chr(12) Then
      oRange.SetRange Start:=oRange.Start, End:=oRange.End - 1
    End If
    Set nDoc = Documents.Add(Template:=ActiveDocument.AttachedTemplate.FullName)
    nDoc.Content.FormattedText = oRange.FormattedText
    nDoc.SaveAs FileName:=Präfix & Format(i, "0000")
    nDoc.Close
  Next i
End Sub
```

(Teilt ein bestehendes Dokument in neue Dokumente auf, wobei aus jeder Seite des bestehenden Dokumentes ein neues Dokument entsteht.)

Tabelle der vordefinierten Textmarken in MS Word

▲ Zurück zum Anfang

Neben den vordefinierten Textmarken mit der Bezeichnung *Line* bzw. *Page*, kennt MS Word noch weitere Marken, welche nach dem selben Prinzip arbeiten. Dabei haben meiner Meinung nach nicht alle vordefinierten Textmarken den selben Stellenwert. Einige davon sind eine echte Bereicherung, weil man Dank ihnen Problemstellungen in den Griff kriegt, welche mit konventionellen Mitteln nicht oder nur schlecht erreichbar sind. Andere wiederum sind als Alternative zu einer konventionellen Lösung zu sehen.

Wo eine Problemlösung mit veritablen Word-Objekten möglich ist, sollte man objektorientiert und nicht mit diesen Textmarken arbeiten.

Die Informationen sind aus der Word-Hilfe (Word2000) entnommen und sind komplett.

Textmarke	Beschreibung	Bemerkungen
\Sel	Aktuelle Markierung oder die Einfügemarke.	Entspricht `Selection.Range`
\PrevSel1	Die letzte Markierung, bei der eine Bearbeitung erfolgte. Wenn Sie zu dieser Textmarke gehen.	Entspricht einem einmaligen Aufruf der **GoBack**-Methode.
\PrevSel2	Die vorletzte Markierung, bei der eine Bearbeitung erfolgte. Wenn Sie zu dieser Textmarke gehen.	Entspricht einem zweimaligen Aufruf der **GoBack**-Methode.

\StartOfSel	Anfang der aktuellen Markierung.	Entspricht
		`Selection.Range.Start`
		bzw.
		`Selection.Collapse wdCollapseStart`

\EndOfSel	Ende der aktuellen Markierung.	Entspricht
		`Selection.Range.End`
		bzw.
		`Selection.Collapse wdCollapseEnd`

\Line	Die aktuelle Zeile bzw. die erste Zeile der aktuellen Markierung. Wenn die Einfügemarke sich am Ende einer Zeile befindet, bei der es sich nicht um die letzte Zeile eines Absatzes handelt, schließt die Textmarke die gesamte nächste Zeile ein.	Zahlreiche Beispiele dazu finden Sie in diesem Skript.

\Char	Aktuelles Zeichen, bei dem es sich um das Zeichen nach der Einfügemarke handelt, wenn nichts markiert ist oder das Zeichen am Anfang der Markierung steht.	Entspricht folgender Mimik:
		`If Selection.Type = 1 Or _`
		`  Selection.Type = 2 Then`
		`  If Selection.Type = 2 Then`
		`  Selection.Collapse wdCollapseStart`
		`  End If`
		`  Selection.Expand Unit:=wdCharacter`
		`End If`

\Para	Aktueller Absatz, bei dem es sich um den Absatz handelt, der die Einfügemarke enthält. Wenn mehrere Absätze markiert sind, handelt es sich um den ersten Absatz der Markierung. Wenn sich die Einfügemarke oder Markierung im letzten Absatz des Dokuments befindet, schließt die Textmarke "\Para" die Absatzmarke nicht ein.	Absätze sind veritable Objekte im Word-Objekt-Modell. Deshalb erübrigt sich eigentlich das Ansprechen über vordefinierte Textmarken. Sie finden in diesem Skript zahlreiche Beispiel, welche den Umgang mit dem Objekt Paragraph erläutern.
\Section	Aktueller Abschnitt, einschließlich des Wechsels am Ende des Abschnitts, falls vorhanden. Der aktuelle Abschnitt enthält die Einfügemarke oder die Markierung. Enthält die Markierung mehrere Abschnitte, ist die Textmarke "\Section" der erste Abschnitt in der Markierung.	Abschnitte sind veritable Objekte im Word-Objekt-Modell. Deshalb erübrigt sich eigentlich das Ansprechen über vordefinierte Textmarken. Um das Section-Objekt anzusprechen, codieren Sie z.B. folgendes: `ActiveDocument.Sections(1).Range.Select`
\Doc	Gesamter Inhalt des aktiven Dokuments, mit Ausnahme der letzten Absatzmarke.	Folgende Code-Zeile markiert den Inhalt des Dokumentes inklusive letzte Absatzmarke: `ActiveDocument.Content.Select`

Textmarke	Beschreibung	Bemerkungen
\Page	Aktuelle Seite, einschließlich des Wechsels am Ende der Seite, falls vorhanden. Die aktuelle Seite enthält die Einfügemarke. Enthält die aktuelle Markierung mehrere Seiten, ist die Textmarke "\Page" die erste Seite der Markierung. Befindet sich die Einfügemarke oder Markierung auf der letzten Seite des Dokuments, enthält die Textmarke "\Page" nicht die letzte Absatzmarke.	Zahlreiche Beispiele dazu finden Sie in diesem Skript.
\StartOfDoc	Der Anfang des Dokuments.	Damit können Sie in einer Schleife, welche das Dokument vom Ende zum Anfang einliest, feststellen, wenn der Anfang erreicht ist.
\EndOfDoc	Das Ende des Dokuments.	Damit können Sie in einer Schleife, welche das Dokument vom Anfang zum Ende einliest, feststellen, wenn das Ende erreicht ist. Ein Beispiel dazu finden Sie in diesem Skript.
\Cell	Aktuelle Zelle in einer Tabelle, bei der es sich um die Zelle handelt, die die Einfügemarke enthält. Sind eine oder mehrere Zellen einer Tabelle in die aktuelle Markierung eingeschlossen, ist die Textmarke "\Cell" die erste Zelle in der Markierung.	Zellen sind veritable Objekte im Word-Objekt-Modell. Deshalb erübrigt sich eigentlich das Ansprechen über vordefinierte Textmarken. Um ein Cell-Objekt anzusprechen, codieren Sie z.B. folgendes: `Selection.Cells(1).Select`
\Table	Aktuelle Tabelle, bei der es sich um die Tabelle handelt, die die Einfügemarke oder die Markierung enthält. Schließt die Markierung mehrere Tabellen ein, ist die Textmarke "\Table" die gesamte erste Tabelle der Markierung. Dies ist auch der Fall, wenn nicht die gesamte Tabelle markiert ist.	Tabellen sind veritable Objekte im Word-Objekt-Modell. Deshalb erübrigt sich eigentlich das Ansprechen über vordefinierte Textmarken. Um ein Table-Objekt anzusprechen, codieren Sie z.B. folgendes: `Selection.Table(1).Select`
\HeadingLevel	Die Überschrift, die die Einfügemarke oder die Markierung mit untergeordneten Überschriften und Text enthält. Handelt es sich bei der aktuellen Markierung um Textkörper, schließt die Textmarke "\HeadingLevel" die vorhergehende Überschrift mit allen Überschriften und Text ein, die der Überschrift untergeordnet sind.	Damit gelingt es Ihnen, ein bestimmtest Kapitel, bzw. Unterkapitel in einem gegliederten Textstück anzusprechen. Um das Kapitel anzusprechen in dem sich die Einfügemarke befindet, verwenden Sie folgenden Code: `Selection.Bookmarks("\HeadingLevel").Select`

Links zu diesem Thema

▲ **Zurück zum Anfang**

Eigene Links

► **Arbeiten mit dem Range-Objekt in MS Word**
► **VBA-technischer Umgang mit Textmarken**

Microsoft Word MVP FAQ Site

► **Determine the index number of the current paragraph, table, section**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/ZS.htm

René Probst, Dezember 2000

Änderungen

- --- nil ---

VBA-technischer Umgang mit Word-Tabellen

◄ Zurück zum Hauptindex

Problem

> Wie funktioniert der programmtechnische Umgang mit Word-Tabellen? Was sind die Möglichkeiten? Wo stellen sich Probleme ein?

Diese Seite

▼ Einleitung
▼ Das Objekt-Modell von Word-Tabellen
▼ Feststellen, ob die verlangte Tabelle im Dokument vorhanden ist
▼ Text aus einer Tabellenzelle auslesen
▼ Word-Tabelle in einem Datenfeld (Array) abbilden
▼ Feststellen in welcher Zelle sich die Einfügemarke befindet
▼ Adresse der Zelle oder Zellbereich ausweisen
▼ Eine bestimmte Tabelle über eine Textmarke auffinden
▼ Eine Tabelle mit VBA erstellen
▼ Zusätzliche Zeilen oder Spalten in eine bestehende Tabelle einbringen
▼ Arbeiten mit Tabellenspalten
▼ Die Range-Eigenschaft einer Tabelle ansprechen
▼ Zellbereich markieren
▼ Werte in Zellen formatieren
▼ Letzte Tabellenzeile jeder Seite auffinden
▼ Formel in eine Zelle der letzen Tabellenzeile einfügen
▼ Verschachtelte Tabellen in Word2000 verarbeiten
▼ Word-Tabelle auf einer UserForm abbilden
▼ Weitere VBA-Splitter im Zusammenhang mit Tabellen
▼ Ein Excel-Tabellenblatt in einer Word-Tabelle abbilden
▼ Eine Access-Tabelle in einer Word-Tabelle abbilden
▼ Links zu diesem Thema

Einleitung

▲ Zurück zum Anfang

Das arbeiten mit Word-Tabellen in einem Programm ist eigentlich eine erfreuliche Sache. Solange die Tabellen nicht ineinander verschachtelt sind, was erst ab Word2000 möglich ist, ist die Handhabung recht einfach, konsistent und einleuchtend. Problematischer wird es, wenn die Tabelle Zellen enthält, welche unter einander verbunden sind oder wenn eine Tabelle vorliegt, welche frei gezeichnet wurde. Viele VBA-Funktionen, wie z.B. aus Auslesen der Zellhöhe, stehen dann nämlich nicht mehr oder nur noch teilweise zur Verfügung.

Erfahrungsgemäss haben Tabellen einen hohen Stellenwert beim Benutzer, das will heissen man greift gerne auf Tabellen zurück. Dieses Skript soll an Hand von konkreten Code-Beispielen aufzeigen, wie man die häufigsten Problemstellungen im Zusammenhang mit Word-Tabellen angehen kann.

Das Objekt-Modell von Word-Tabellen

▲ Zurück zum Anfang

Das Objekt-Modell von Tabellen birgt keine grossen Überraschungen; es ist ziemlich einleuchtend aufgebaut. Hier einige wichtigen Grundsätze des *Table-Objektes* in MS Word:

- Ein Word-Dokument kann **0 - n** Tabellen beinhalten. Jeder dieser Tabelle kann über den Index angesprochen werden. Dieser Index ist sequentiell und aufsteigend. Einen aussagekräftigen Namen besitzt eine Tabelle nicht und man kann auch keinen vergeben, es sei denn indirekt über eine Textmarke, welche man an beliebiger Stelle in einer Tabelle platziert.
- Eine Tabelle besteht aus **1-63 Spalten**, welche Unterobjekte der Tabelle darstellen. Diese Objekte werden in VBA mit ***Columns*** angesprochen. Die Zählung ist strikt numerisch und beginnt mit 1.
- Im weiteren besteht eine Tabelle aus **1-n Zeilen**, wobei n fast beliebig gross sein kann. Auch diese Objekte in Word-VBA ***Rows*** genannt, sind Elemente der Tabelle . Die Zählung ist strickt numerisch und beginnt mit 1.
- Das kleinste Element der Tabelle schliesslich ist die Zelle. Diese werden in VBA mit ***Cells*** angesprochen. VBA erweist sich als flexibel können Zellen doch sowohl als Elemente von Zeilen, Spalten oder sogar als Elemente von Tabellen direkt angesprochen werden. Die Syntax ist allerdings unterschiedlich. Spricht man eine Zelle über eine Zeile oder eine Spalte an, dann hat die Zelle eine Sequenznummer; werden Zellen dagegen direkt als Elemente von Tabellen angesprochen, dann muss man deren Koordinaten angeben. Die Zählung ist strickt numerisch. Die erste Zelle ist die Zelle, welche oben links in der Tabelle thront, dann geht die Zählung in Leserichtung weiter, also von links nach rechts und dann von oben nach unten.
- Zeilen, Spalten wie auch Zellen verfügen über ein **Range-Objekt.** Damit ist der Zugriff einfach und effizient. Bei der Rückgabe des *Cell-Ranges* ist zu beachten, dass der Text in diesem Range am Ende immer zwei Steuerzeichen aufweist (ASCII-Steuerzeichen dezimal 13 und 9). Diese Steuerzeichen sollte man abtrennen, damit der Text in sauberer Form vorliegt.
- Bevor auf eine Tabelle zugegriffen wird, sollte immer überprüft werden, ob die entsprechende Tabelle im Dokument wirklich vorhanden ist, sonst bricht der Code mit einer Fehlermeldung ab. Das selbe gilt für Zeilen und Spalten oder Zellen.

Feststellen, ob die verlangte Tabelle im Dokument vorhanden ist

▲ Zurück zum Anfang

Ermitteln, ob das Dokument Tabellen enthält

Im folgenden wird ermittelt, ob sich im Dokument eine Tabelle befindet. Wenn nicht wird die Verarbeitung abgebrochen.

```
Sub PruefenObTabelleVorhanden()
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument enthält keine Tabelle.", vbInformation
    Exit Sub
  End If
  ActiveDocument.Tables(1).Select
End Sub
```

(Überprüft, ob das Dokument eine Tabelle enthält.)

Ermitteln, ob sich die Einfügemarke in einer Tabelle befindet

Im folgenden wird ermittelt, ob sich die Einfügemarke in einer Tabelle befindet. Wenn nicht wird die Verarbeitung abgebrochen.

```
Sub PruefenObSichDieEinfuegemarkeInEinerTabelleBefindet()
  If Selection.Information(wdWithInTable) = False Then
    MsgBox "Die Einfügemarke befindet sich nicht in einer Tabelle.", vbInformation
    Exit Sub
  End If
  Selection.Tables(1).Select
End Sub
```

(Überprüft, ob die Einfügemarke in einer Tabelle steht.)

Text aus einer Tabellenzelle auslesen

▲ Zurück zum Anfang

Wie schon Eingangs erwähnt, können Sie eine Zelle direkt über das Tabellenobjekt oder aber über die Zeile bzw. die Spalten, welche diese Zelle beherbergt auslesen.

Beachten Sie, dass der Text in der Zelle immer mit zwei Steuerzeichen endet, welche abgetrennt werden sollen.

Im Folgenden wird der Text der Zelle, welche sich in der 3. Zeile und in der 2. Spalte befindet (man könnte diese Zelle gedanklich auch mit **B3** adressieren) ausgelesen. Sie finden nun alle drei Methoden, um diese Problemstellung anzugehen.

```
Sub TabellenzelleAuslesen1()
  strText = ActiveDocument.Tables(1).Cell(3, 2).Range.Text
  strText = Left(strText, Len(strText) - 2)
  MsgBox strText
End Sub
```

(Die Zelle mit der Adresse *B3* wird ausgelesen. Dabei wird die Zelle direkt als Element der Tabelle angesprochen.)

```
Sub TabellenzelleAuslesen2()
  strText = ActiveDocument.Tables(1).Rows(3).Cells(2).Range.Text
  strText = Left(strText, Len(strText) - 2)
  MsgBox strText
End Sub
```

(Die Zelle mit der Adresse *B3* wird ausgelesen. Dabei wird diese Zelle als Element der dritten Tabellenzeile angesprochen.)

```
Sub TabellenzelleAuslesen3()
  strText = ActiveDocument.Tables(1).Columns(2).Cells(3).Range.Text
  strText = Left(strText, Len(strText) - 2)
  MsgBox strText
End Sub
```

(Die Zelle mit der Adresse *B3* wird ausgelesen. Dabei wird die Zelle als Element der zweiten Tabellenspalte angesprochen.)

Um sicher zu gehen, dass der Code nicht mit einer Fehlermeldung abbricht, müssten Sie eigentlich noch prüfen, ob die Tabelle und die zu adressierenden Zelle wirklich vorhanden sind. Im Folgenden sehen Sie den Code des ersten Beispieles nochmals, diesmal aber mit der erwähnten Validierung.

```
Sub TabellenzelleAuslesenMitValidierung()
  Dim oTable As Table
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument enthält keine Tabelle.", vbInformation
    Exit Sub
  End If
  Set oTable = ActiveDocument.Tables(1)
  If oTable.Rows.Count < 3 Or oTable.Columns.Count < 2 Then
    MsgBox "Die verlangte Zelle existiert nicht.", vbInformation
    Exit Sub
  End If
  strText = ActiveDocument.Tables(1).Cell(3, 2).Range.Text
  strText = Left(strText, Len(strText) - 2)
  MsgBox strText
End Sub
```

(Zuerst wird sichergestellt, dass die verlangte Tabelle und die verlangten Zelle tatsächlich existieren. Anschliessend wird der Text der Zelle *B3* zurückgegeben.)

Word-Tabelle in einem Datenfeld (*Array*) abbilden

▲ Zurück zum Anfang

Eine Word-Tabelle lässt sich sehr einfach in einem Datenfeld (*Array*) abbilden. Am geeignetsten dazu ist ein zweidimensionaler Array. Dabei stellt die erste Dimension die einzelnen Tabellenzeilen und die zweite Dimension die einzelnen Tabellenspalten dar. Selbstverständlich ist das auch umgekehrt möglich.

Im Folgenden finden Sie zuerst eine allgemeingültige Routine um eine Tabelle in einem eindimensionalen Datenfeld abzubilden. Dabei wird mit der Zelle oben links in der Tabelle begonnen. Anschliessend folgen die nächsten Zellen in Leserichtung, also von links nach rechts und dann von oben nach unten.

Eindimensionales Datenfeld (*Array*)

```
Sub WordTabelleNachEindimensionalenArray1()
  Dim x() As String
  Dim oTable As Table
  Dim Zelle As Cell
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument enthält keine Tabelle.", vbInformation
    Exit Sub
  End If
  Set oTable = ActiveDocument.Tables(1)
  ReDim x(1 To oTable.Range.Cells.Count)
  i = 0
  For Each Zelle In oTable.Range.Cells
    i = i + 1
    x(i) = Left(Zelle.Range.Text, Len(Zelle.Range.Text) - 2)
  Next Zelle
End Sub
```

(Eine Word-Tabelle wird nach einem eindimensionalen Array ausgelesen. Die Leserichtung in diesem Beispiel lautet: A1, B1, C1, A2, B2, C2, A3, B3, C3, etc.)

Im Folgenden wird nochmals ein eindimensionaler *Array* der ersten Tabelle im Dokument gebildet. Dieses Mal wird die Reihenfolge der Zellen aber anders eingelesen, nämlich von oben nach unten und dann von rechts nach links, also nach dem Muster A1, A2, A3, B1, B2, B3, C1, C2, C3, etc.

```
Sub WordTabelleNachEindimensionalenArray2()
  Dim x() As String
  Dim oTable As Table
  Dim Spalte As Column
  Dim Zelle As Cell
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument enthält keine Tabelle.", vbInformation
    Exit Sub
  End If
  Set oTable = ActiveDocument.Tables(1)
  ReDim x(1 To oTable.Range.Cells.Count)
  i = 0
  For Each Spalte In oTable.Columns
    For Each Zelle In Spalte.Cells
      i = i + 1
      x(i) = Left(Zelle.Range.Text, Len(Zelle.Range.Text) - 2)
    Next Zelle
  Next Spalte
End Sub
```

(Eine Word-Tabelle wird nach einem eindimensionalen Array ausgelesen. In diesem Beispiel werden die Zellen in folgender Reihenfolge ausgelesen: A1, A2, A3, B1, B2, B3, C1, C2, C3, etc.)

Zweidimensionales Datenfeld (*Array*)

Geeigneter als ein eindimensionales Datenfeld um eine Tabelle auszulesen, ist aber ein zweidimensionaler *Array*. Im folgenden Beispiel wird eine allgemeingültige Routine aufgezeigt, welche diese Problemstellung löst.

Im ersten Beispiel stellt die erste Dimension die Zeilen und die zweite Dimension die Spalten dar. Im zweiten Beispiel wird die Tabelle dann *transponiert*, d.h. die Spalten stellen die erste Dimension und die Zeilen die zweite Dimension dar.

```
Sub WordTabelleNachZweidimensionalenArray1()
  Dim x() As String
  Dim oTable As Table
  Dim Zeile As Row
  Dim Zelle As Cell
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument enthält keine Tabelle.", vbInformation
    Exit Sub
  End If
  Set oTable = ActiveDocument.Tables(1)
  ReDim x(1 To oTable.Rows.Count, 1 To oTable.Columns.Count)
  i = 0
  For Each Zeile In oTable.Rows
    i = i + 1
    j = 0
    For Each Zelle In Zeile.Cells
      j = j + 1
      x(i, j) = Left(Zelle.Range.Text, Len(Zelle.Range.Text) - 2)
    Next Zelle
  Next Zeile
End Sub
```

(Eine Word-Tabelle wird in nach einem zweidimensionalen Array ausgelesen. In diesem Beispiel stellen die Zeilen die erste Dimension und die Spalten die zweite Dimension dar.)

Nun wird die Tabelle also transponiert, so dass die Spalten die erste Dimension und die Zeilen die zweite Dimension darstellen.

```
Sub WordTabelleNachZweidimensionalenArray2()
  Dim x() As String
  Dim oTable As Table
  Dim Spalte As Column
  Dim Zelle As Cell
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument enthält keine Tabelle.", vbInformation
    Exit Sub
  End If
  Set oTable = ActiveDocument.Tables(1)
  ReDim x(1 To oTable.Columns.Count, 1 To oTable.Rows.Count)
  i = 0
  For Each Spalte In oTable.Columns
    i = i + 1
    j = 0
    For Each Zelle In Spalte.Cells
      j = j + 1
      x(i, j) = Left(Zelle.Range.Text, Len(Zelle.Range.Text) - 2)
    Next Zelle
  Next Spalte
End Sub
```

(Eine Word-Tabelle wird in nach einem zweidimensionalen Array ausgelesen. In diesem Beispiel stellen die Spalten die erste Dimension und die Zeilen die zweite Dimension dar.)

Feststellen in welcher Zelle sich die Einfügemarke befindet

▲ Zurück zum Anfang

Im Folgenden finden Sie die Mimik um festzustellen, in welcher Zelle sich die Einfügemarke befindet. Ist ein Zellbereich markiert, dann wird das Ende der Markierung zurückgegeben.

```
Sub FeststellenInWelcherZelleSichDieEinfügemarkeBefindet()
  If Selection.Information(wdWithInTable) = False Then
    MsgBox "Die Einfügemarke befindet sich nicht in einer Tabelle.", vbInformation
    Exit Sub
  End If
  x = Selection.Information(wdEndOfRangeRowNumber)
  y = Selection.Information(wdEndOfRangeColumnNumber)
  MsgBox "Die Einfügemarke befindet sich:" & vbCrLf & _
     "in der Zeile: " & x & vbCrLf & "und in der Spalte: " & y
End Sub
```

(Feststellen in welcher Zelle sich die Einfügemarke befindet.)

Adresse der Zelle oder Zellbereich ausweisen

▲ Zurück zum Anfang

Es geht auch etwas schöner, wie das folgende Beispiel beweist. Die folgende Routine benutzt die Notation, welche wir aus Excel kennen, um die "aktive" Zelle, bzw. den markierten Zellbereich zurückzugeben. Die Routine erweist sich im Umgang mit Word-Tabellen immer wieder als ausserordentlich praktisch. Die Zellbezeichnung bzw. der Zellbereich wird nach der Ausführung des Makros in der Statuszeile angezeigt.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Titel = "Tabellen Position"

Sub TabellenPositionAnzeigen()
   If Selection.Information(wdWithInTable) = False Then
      Markierung = 0
      If Selection.Type = 1 Then
        Meldung = "Der Zeiger befindet sich nicht in einer Tabelle!"
      Else
        Meldung = "Die Markierung stellt keinen gültigen Tabellenbereich dar!"
      End If
   Else
      If Selection.Type = 1 Then
         Position = TabellenPosition
         Markierung = 1
         tmp = Position
         Meldung = "Die aktive Zelle heisst: " & tmp & "."
      Else
         Bereich = TabellenBereich
         Markierung = 2
         tmp = Bereich
         Meldung = "Der markierte Tabellenbereich heisst: " & tmp & "."
      End If
   End If
   'MsgBox Meldung, vbInformation, Titel
   StatusBar = Meldung
End Sub

Private Function TabellenPosition() As String
  TabellenPosition = _
     AlphaNum(Selection.Information(wdStartOfRangeColumnNumber)) & _
        Selection.Information(wdStartOfRangeRowNumber)
End Function

Private Function TabellenBereich() As String
   TabellenBereich = _
      AlphaNum(Selection.Information(wdStartOfRangeColumnNumber)) & _
         Selection.Information(wdStartOfRangeRowNumber) & " - " & _
            AlphaNum(Selection.Information(wdEndOfRangeColumnNumber)) & _
               Selection.Information(wdEndOfRangeRowNumber)
End Function

Private Function AlphaNum(Num As Long) As String
  Z = 0
  While Num > 26
    Z = Z + 1
    Num = Num - 26
  Wend
  AlphaNum = Chr(Num + 64)
  If Z > 0 Then AlphaNum = Chr(Z + 64) & AlphaNum
End Function
```

(Dieser Code teilt dem Benutzer mit, in welcher Zelle sich die Einfügemarke befindet Sind mehrere Zellen markiert, wird die Start- und Endzelle ausgewiesen.)

Eine bestimmte Tabelle über eine Textmarke auffinden

▲ Zurück zum Anfang

Das Auffinden einer Tabelle über deren Index erweist sich nicht immer als praxisgerecht, da sich dieser Index ja sehr schnell ändern kann, wenn das Dokument mit zusätzlichen Tabellen bereichert wird bzw. Tabellen gelöscht werden.

Wie schon Eingangs erwähnt, besitzt eine Tabelle keinen (eindeutigen) Namen und Sie können auch keinen solchen vorgeben.

Der Ausweg aus diesem Dilemma ist eine Textmarke, welche Sie an beliebiger Stelle in der Tabelle vergeben können. Es ist dann sehr einfach mit VBA auf genau diese Tabelle zuzugreifen.

Angenommen es gibt eine Tabelle, welche eine Textmarke mit dem eindeutigen Name **Jahresabrechnung** aufweist, dann können Sie diese Tabelle wie folgt ansprechen:

```
Sub TabelleUeberTextmarkeAnsprechen()
  Dim oTable As Table
  If ActiveDocument.Bookmarks.Exists("Jahresabrechnung") Then
    Set oTable = _
       ActiveDocument.Bookmarks("Jahresabrechnung").Range.Tables(1)
    oTable.Select
  End If
End Sub
```

(Eine Tabelle wird über eine Textmarke, welche sich in der entsprechenden Tabelle befindet angesprochen..)

Eine Tabelle mit VBA erstellen

▲ Zurück zum Anfang

Das Erstellen einer Tabelle mittels VBA ist denkbar einfach, der Vorgang lässt sich in der Regel auch problemlos aufzeichnen. Im Folgenden wird an der Stelle der Einfügemarke eine kleine Tabelle erstellt.

```
Sub TabelleEinfügen1()
  With Selection
    If .Information(wdWithInTable) = True Then
      .MoveDown Count:=.Tables(1).Rows.Count - _
         .Information(wdEndOfRangeRowNumber) + 2
    End If
    ActiveDocument.Tables.Add Range:=.Range, NumRows:=2, NumColumns:=2
  End With
End Sub
```

(Eine Tabelle wird in das Word-Dokument eingebracht. Sollte sich an der Stelle, wo die Einfügemarke ist, bereits eine Tabelle befinden, wird ausgerechnet, um wie viele Zeilen der Zeiger nach unten hüpfen muss.)

Hier nochmals die selbe Aufgabenstellung, etwas eleganter gelöst.

```
Sub TabelleEinfügen2()
  Dim oRange As Range
  Set oRange = Selection.Range
  If Selection.Information(wdWithInTable) = True Then
    oRange.SetRange Start:=Selection.Tables(1).Range.End + 1, _
                    End:=Selection.Tables(1).Range.End + 1
  End If
  ActiveDocument.Tables.Add Range:=oRange, NumRows:=2, NumColumns:=2
End Sub
```

(Verfolgt das selbe Ziel, wie das obige Beispiel. Diesmal wird aber mit einem Range-Objekt gearbeitet. Diese Lösung ist der obigen vorzuziehen.)

Im Folgenden wird eine kleine Tabelle erstellt. Anschliessend werden die Zellen der Tabellen mit Text gefüllt.

```
Sub TabelleErstellen()
  Dim oTable As Table, Zelle As Cell
  If Selection.Information(wdWithInTable) = True Then
    MsgBox "Die Einfügemarke befindet bereits in einer Tabelle.", vbInformation
    Exit Sub
  End If
  Set oTable = Selection.Tables.Add(NumRows:=2, NumColumns:=2,
Range:=Selection.Range)
  i = 0
  For Each Zelle In oTable.Range.Cells
    i = i + 1
    Zelle.Range.Text = "Das ist die " & i & ". Zelle"
  Next
End Sub
```

(Eine Tabelle wird mit VBA erstellt. Die Tabelle wird am Ort der Einfügemarke generiert. Anschliessend werden die Zellen dieser Tabelle mit Daten gefüllt.)

! Bei grösseren und oder komplexeren Tabellen ist es äusserst ineffizient zuerst eine Tabelle zu erstellen und anschliessend diese Tabelle mit Daten zu füllen. Um Dimensionen schneller ist es, zuerst die Daten zusammenzusetzen und dann diese Daten in eine Tabelle umzuwandeln. Die VBA-Methode mit dem Namen ***ConvertToTable*** eignet sich dazu hervorragend und ist einfache anzuwenden, wie das folgende Beispiel aufzeigt.

☛ Für **Word2000** gilt, dass sich die Verarbeitungsgeschwindigkeit von Tabellen mit der Installation des Office *Service Releases 1* verbessern soll.

Ziel des Codes ist es, Schriftproben für alle Schriften zu erstellen, welche unter MS Word zur Verfügung stehen. Dafür wird eine zweispaltige Tabelle erzeugt, wobei in der linken Spalte der Schriftname erscheint und in der rechten Spalte ein Beispieltext, welcher mit der Schriftart in der linken Spalte formatiert ist. Die Routine vollzieht folgende Schritte:

1. Es wird ein neues Dokument erstellt, welches die Tabelle aufnehmen soll.
2. Die Schriften werden zusammengesucht. Abwechslungsweise wird der Schriftname und dann der Beispieltext in einer Zeichenfolge verkettet. Als Trennzeichen für die Spalten dient ein Tabulator. Als Steuerzeichen für eine neue Zeile dient ein Zeilenumbruch.
3. Die so konstruierte Zeichenfolge wird in das Dokument geschrieben.
4. Der entstandene Text wird mittels *ConvertToTable-Methode* in eine Tabelle umgewandelt.
5. Die rechte Spalte, welches den Beispieltext enthält, wird mit der entsprechenden Schriftart formatiert.

Kopieren Sie den folgenden Code in ein neues Modul und testen Sie das Makro.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Probetext = _
   "Franz jagt im komplett verwahrlosten Taxi quer durch Bayern. 0123456789"

Sub SchriftProbeErstellen()
  For Each Schrift In Application.FontNames
    strText = strText & Schrift & vbTab & Probetext & vbCrLf
  Next
  Dim nDoc As Document
  Set nDoc = Documents.Add
  nDoc.Content = "Schriftart" & vbTab & "Probetext" & vbCrLf & strText
  nDoc.Content.ConvertToTable Separator:=vbTab, _
     Format:=wdTableFormatProfessional, AutoFit:=True
  Application.ScreenUpdating = False
  Dim Zeile As Row
  For Each Zeile In nDoc.Tables(1).Rows
    If Zeile.Index > 1 Then
      Zeile.Range.Cells(2).Range.Font.Name = _
         Left(Zeile.Cells(1).Range.Text, _
            Len(Zeile.Cells(1).Range.Text) - 2)
      StatusBar = Zeile.Index - 1 & "Schriften formatiert - Bitte warten..."
    End If
  Next
  StatusBar = Application.FontNames.Count & "Schriften verfügbar."
  Application.ScreenUpdating = True
End Sub
```

(Erstellt eine Schriftprobe für jede Schriftart, welche unter Word zur Verfügung steht. Da das Arbeiten mit grossen Tabellen sehr langsam sein kann, sollten Sie die Tabelle so spät wie möglich generieren.)

Das letzte Beispiel zeigt eines sehr schön: die Routine läuft sehr schnell, solange nicht mit der Tabelle selbst gearbeitet wird. Auch die *ConvertToTable-Methode* ist ordentlich schnell. Der Schluss, in welchem eine Word-Tabelle nachformatiert wird, geht dann aber schleppend voran.

Es ist also wichtig, möglichst lange ohne Tabelle zu arbeiten. Im Beispiel wird eine String-Variable verwendet, um das Resultat vorzubereiten. Eine solche Variable kann bis zu 64kb Daten aufnehmen. Das entspricht immerhin einen Equivalent von etwa 12 - 20 Seiten Text.

Sollten Sie begründete Befürchtungen haben, dass 64kb für Ihre Aufgabenstellung nicht ausreicht, dann können Sie wenigstens die einzelnen Zeilen Ihrer zukünftigen Tabelle in einer String-Variable vorbereiten. Diese Zeilen schreiben Sie dann fortwährend in das Dokument. Zum Schluss wandeln Sie dann diesen Text in eine Tabelle um.

Um im Text die zukünftigen Spalten anzudeuten, können Sie das Tab-Steuerzeichen verwenden. Wenn Sie annehmen müssen, dass der Inhalt der Zellen selbst solche Tab-Steuerzeichen enthalten könnte, dann können Sie problemlos auf ein anderes nicht druckbares Zeichen ausweichen. Nicht jedes Zeichen eignet sich dafür. Verwenden Sie z.B. das ASCII-Zeichen *dezimal 1* wie im folgenden Beispiel und reflektieren Sie diese Entscheidung in der *ConvertToTable-Methode*.

Im folgenden Beispiel werden die Dateien eines Verzeichnisses auf Ihrem PC aufgelistet. Die Namen dieser Dateien werden mit dateispezifischen Informationen ergänzend. Es wird dann eine Tabelle erstellt, welche Dateinamen und diese zusätzlichen Informationen sortiert nach Dateigrösse enthält. Der programmtechnische Ablauf ist wie folgt:

1. Die Dateien eines vorgegebenen Verzeichnisses werden ausgelesen und mit dateispezifischen Informationen ergänzt.
2. In einer String-Variablen wird immer genau eine Zeile zusammengesetzt. Als Trennzeichen (Delimiter) für die Spalten dient das ASCII-Steuerzeichen *dezimal 1*. Als Zeilenabschluss dient eine Absatzmarke (ASCII-Steuerzeichen *dezimal 13)*. Diese String-Variable wird dann in das Dokument geschrieben.
3. Wenn alle Dateien des Verzeichnisses in genannter Weise abgearbeitet sind, wird den Daten noch die Kopfinformation beigefügt. Nun wird eine Tabelle erstellt. Diese Tabelle wird dann mit der *Sort-Methode* sortiert.
4. Einzelne Spalten der Tabelle werden nun noch nachformatiert, damit die Tabelle an Lesbarkeit gewinnt.

Beachten Sie, dass die Mimik um die Dateien zusammenzusuchen nicht rekursiv ist. Es werden also ausschliesslich Dateien gefunden, welche sich direkt im vorgegebenen Verzeichnis befinden, Unterverzeichnisse dagegen werden nicht durchsucht.

Wenn Sie ein Verzeichnis vorgeben, welche viele Dateien enthält, dann kann sich das Ergebnis über viele Seiten erstrecken. Spätestens dann werden Sie einmal mehr erkennen können, dass das Zusammenstellen der Daten und die Generierung der Tabelle schnell abläuft, die Nachformatierung aber, die mit dem Tabellenobjekt gemacht werden muss, ist ein schleppender Vorgang.

Die Formatierung der Spalte *Dateigrösse* muss nachträglich erfolgen, damit die *Table-Sort-Methode* zu einem korrekten Ergebnis führt. Durch diesen zusätzlichen Vorgang verlängert sich die Laufzeit der Routine um ein zigfaches.

```
Private Const Verzeichnis = "C:\Windows"

Sub DateienAuflisten()
  Application.ScreenUpdating = False
  Dim nDoc As Document
  Set nDoc = Documents.Add
  'Dateien auslesen
  x = Dir(Verzeichnis & "\", vbArchive + vbHidden + vbReadOnly)
  While Not x = ""
    nDoc.Content.InsertAfter x & Chr(1) & GroesseAuslesen(x) & Chr(1) & _
```

```
        DatumAuslesen(x) & Chr(1) & AttributeAuslesen(x) & vbCrLf
    x = Dir
  Wend
  'Schriftgard vorgeben und Abstände
  nDoc.Content.Font.Size = 9
  nDoc.Content.ParagraphFormat.SpaceBefore = 2
  nDoc.Content.ParagraphFormat.SpaceAfter = 2
  'Kopfinformation der Tabelle im Dokument einfügen
  nDoc.Content.InsertBefore "Dateiname" & Chr(1) & "Grösse" & Chr(1) & _
     "Datum" & Chr(1) & "Schreibgeschützt" & Chr(1) & "Versteckt" & vbCrLf
  'Text in Tabelle konvertieren
  StatusBar = "Die Tabelle wird jetzt erstellt."
  Dim oTable As Table, Zelle As Cell
  Set oTable = nDoc.Content.ConvertToTable(Separator:=Chr(1), AutoFit:=True)
  'Tabelle nach Dateigrösse sortieren
  oTable.Sort ExcludeHeader:=True, FieldNumber:=2, _
     SortFieldType:=wdSortFieldNumeric, SortOrder:=wdSortOrderDescending
  'Feld Dateigrösse formatieren
  StatusBar = _
     "Die Tabelle wird nun nachformatiert. Das nimmt einige Zeit in Anspruch."
  Assistant.Animation = msoAnimationWorkingAtSomething
  System.Cursor = wdCursorWait
  For Each Zelle In oTable.Columns(2).Cells
    If Zelle.RowIndex > 1 Then
      tStr = Left(Zelle.Range.Text, Len(Zelle.Range.Text) - 2)
      tStr = Format(tStr, "###'###'##0")
      ofs = InStr(tStr, "'")
      While ofs = 1 'Strip leading quotes
        tStr = Right(tStr, Len(tStr) - 1)
        ofs = InStr(tStr, "'")
      Wend
      Zelle.Range.Text = tStr
      Zelle.Range.Font.Name = "Courier New"
      Zelle.Range.ParagraphFormat.Alignment = wdAlignParagraphRight
    End If
  Next
  'Textausrichtung in Spalten 4 + 5 ändern
  nDoc.Range(Start:=oTable.Cell(2, 4).Range.Start, _
    End:=oTable.Cell(oTable.Rows.Count, 5).Range.End).Select
  Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
  'Kopf der Tabelle auf jeder Seite wiederholen
  oTable.Rows(1).HeadingFormat = True
  'Zum Anfang im Dokument springen
  nDoc.Range(0, 0).Select
  StatusBar = "Die Liste ist nun erstellt."
  System.Cursor = wdCursorNormal
  Assistant.Animation = msoAnimationCharacterSuccessMajor
  Application.ScreenUpdating = True
End Sub

Private Function GroesseAuslesen(ByVal Datei As String) As Long
  GroesseAuslesen = FileLen(Verzeichnis & "\" & Datei)
End Function
```

```
Private Function DatumAuslesen(ByVal Datei As String) As Date
  DatumAuslesen = FileDateTime(Verzeichnis & "\" & Datei)
End Function

Private Function AttributeAuslesen(ByVal Datei As String) As String
  Attr = GetAttr(Verzeichnis & "\" & Datei)
  If Attr >= 32 Then Attr = Attr - 32 'Archive
  Select Case Attr
    Case 0: t = "-" & Chr(1) & "-"
    Case 1: t = "X" & Chr(1) & "-"
    Case 2: t = "-" & Chr(1) & "X"
    Case 3: t = "X" & Chr(1) & "X"
  End Select
  AttributeAuslesen = t
End Function
```

(Eine Word-Tabelle mit umfangreichen Daten wird gänzlich mittels VBA erstellt. Im Gegensatz zum vorhergehenden Beispiel, erfolgt der Aufbau hier Zeile um Zeile, welche fortwährend in das Dokument geschrieben werden. Dann werden diese Zeilen in eine Tabelle umgewandelt. Die Tabelle wird anschliessend noch nachformatiert.)

Zusätzliche Zeilen oder Spalten in eine bestehende Tabelle einbringen

▲ Zurück zum Anfang

Um eine bestehende Tabelle zu vergrössern können Sie wie folgt vorgehen.

Zusätzliche Zeilen einbringen

Das folgende Beispiel ergänzend eine bestehend Tabelle - unten - mit einer zusätzlichen Zeile.

```
Sub TabelleUntenUmEineZeileErweitern()
  Dim oTable As Table, Zeile As Row
  Set oTable = ActiveDocument.Tables(1)
  Set Zeile = oTable.Rows.Add
End Sub
```

(Eine zusätzliche Zeile wird unten an eine Tabelle angehängt.)

Im folgenden Beispiel wird die Tabelle um eine Zeile erweitert. Diese Zeile wird am Anfang der Tabelle eingefügt.

```
Sub TabelleObenUmEineZeileErweitern()
  Dim oTable As Table, Zeile As Row
  Set oTable = ActiveDocument.Tables(1)
  Set Zeile = oTable.Rows.Add(BeforeRow:=oTable.Rows(1))
End Sub
```

(Eine zusätzliche Zeile wird oben an eine Tabelle angehängt.)

Im folgenden Beispiel wird eine Tabelle nach unten mit einer Spalte erweitert. Zusätzlich wird in die erste Zelle dieser neuen Spalte eine Textmarke eingebracht.

```
Sub TabelleUntenUmEineZeileErweiternUndTextmarkeEinfuegen()
  Dim oTable As Table, Zeile As Row
  Set oTable = ActiveDocument.Tables(1)
  Set Zeile = oTable.Rows.Add
  ActiveDocument.Bookmarks.Add Name:="Kurs" & Format(Zeile.Index, "000"), _
     Range:=ActiveDocument.Range(Zeile.Range.Start, Zeile.Range.Start)
End Sub
```

(Eine zusätzliche Zeile wird unten an eine Tabelle angehängt. In die erste Zelle dieser neuen Zeile wird dann eine Textmarke eingefügt.)

Zusätzliche Spalten einbringen

Es folgen nun noch zwei analoge Beispiele diesmal aber für Spalten.

```
Sub TabelleRechtsUmEineSpalteErweitern()
  Dim oTable As Table, Spalte As Column
  Set oTable = ActiveDocument.Tables(1)
  Set Spalte = oTable.Columns.Add
End Sub
```

(Eine zusätzliche Spalte wird rechts an eine Tabelle angehängt.)

```
Sub TabelleLinksUmEineSpalteErweitern()
  Dim oTable As Table, Spalte As Column
  Set oTable = ActiveDocument.Tables(1)
  Set Spalte = oTable.Columns.Add(BeforeColumn:=oTable.Columns(1))
End Sub
```

(Eine zusätzliche Spalte wird links an eine Tabelle angehängt.)

Arbeiten mit Tabellenspalten

▲ Zurück zum Anfang

Im folgenden Beispiel wird in der dritten Spalte der ersten Tabelle im Hauptteil des Dokumentes der Ortsbegriff *Basel* durch das französischsprachige Equivalent *Bâle* ersetzt. Da die Zellen einer Spalte keinen zusammenhängenden Bereich darstellen (wenn man es in der üblichen Leserichtung betrachtet), behandeln wir die Spalte, als eine Sammlung von Zellen, welche wir mit einer Schleife bedienen können. Das Objekt *Columns* gibt es zwar in VBA, allerdings verfügt es nicht über die *Range*-Eigenschaft. Diese Tatsache sollte uns nicht erstaunen, denn eine Spalte hat im Sinne des Word-Objekt-Models kein klarer Start- bzw. Endpunkt; sie liegt quer im Felde. Würde man die Spalte in Leserichtung anmalen, müsste der Pinsel immer wieder von neuem angesetzt bzw. abgesetzt werden.

```
Sub TabellenspalteDurchsuchen()
  Application.ScreenUpdating = False
  Dim oTable As Table, Zelle As Cell
  Set oTable = ActiveDocument.Tables(1)
  For Each Zelle In oTable.Columns(3).Cells
    With Zelle.Range.Find
      .Text = "Basel"
      .Replacement.Text = "Bâle"
      .Execute Replace:=wdReplaceAll
    End With
  Next
  Application.ScreenUpdating = True
End Sub
```

(Die Routine demonstriert, wie eine Tabellenspalte durchsucht werden kann.)

Die *Range*-Eigenschaft einer Tabelle ansprechen

▲ Zurück zum Anfang

Im folgenden Beispiel soll vor einer bestehenden Tabelle eine Kapitelüberschrift eingefügt werden. Das ist gar nicht so einfach zu lösen, speziell dann nicht, wenn das Dokument unmittelbar mit einer Tabelle beginnt.

Der folgende Code ist allgemeingültig. Er löst diese Aufgabe auch dann, wenn die Tabelle das Erste ist, was im Dokument steht und damit erst Platz am Anfang geschaffen werden muss.

```
Sub VorDerTabelleEineUeberschriftSchreiben()
  Dim oTable As Table, oRange As Range, sRange As Range
  Set oTable = ActiveDocument.Tables(1)
  Set oRange = oTable.Range
  If oRange.Start = 0 Then
    Set sRange = Selection.Range
    ActiveDocument.Range(0, 0).Select
    Selection.SplitTable
    Selection.Range = sRange
  End If
  oRange.SetRange Start:=oRange.Start - 1, End:=oRange.Start - 1
  oRange.InsertBefore ("Periodensystem der chemischen Elemente")
  oRange.Style = wdStyleHeading3
End Sub
```

(Vor einer bestehenden Tabelle, wird eine Kapitelüberschrift gesetzt.)

Im folgenden Code-Beispiel wird die Höhe (in Millimeter) einer Tabelle bestimmt.

```
Sub HoeheEinerTabelleBestimmen()
  Dim oTable As Table
  Set oTable = ActiveDocument.Tables(1)
  With ActiveDocument
    Hoehe = PointsToMillimeters(.Range(Start:=oTable.Range.End, _
       End:=oTable.Range.End).Information(wdVerticalPositionRelativeToPage) - _
          .Range(Start:=oTable.Range.Start, _
             End:=oTable.Range.Start).Information(wdVerticalPositionRelativeToPage))
  End With
  strT = "Die Höhe der Tabelle beträgt " & vbCrLf & Format(Hoehe, "###.0") & " mm"
  MsgBox strT
End Sub
```

(Die Höhe einer Tabelle wird bestimmt.)

Zellbereich markieren

▲ Zurück zum Anfang

Mitunter taucht die Frage auf, wie man einen ganz bestimmten Zellbereich in einer Word-Tabelle markieren kann, um dann diesen Bereicht auf eine bestimmte Art und Weise zu formatieren.

Um dies zu erreichen, ist es wohl das geschickteste zuerst einen *Range* zu definieren. Da jeder Zellinhalt selbst als ein *Range* angesehen werden kann, ist es möglich einen Bereich zu definieren, welcher zusammenhängende Zellen beinhaltet.

Im folgenden Beispiel soll der Text in den Spalten 2+3 rechtsbündig ausgerichtet werden. Die erste Zeile, welche die Kopfinformation der Tabelle darstellt, soll davon aber nicht betroffen sein. Konkret soll also der Zellbereich **B2 - Cx** markiert werden, wobei x für die letzte Zeile der Tabelle steht.

```
Sub ZellbereichMarkieren()
  Dim oTable As Table, oRange As Range
  Set oTable = ActiveDocument.Tables(1)
  Set oRange = ActiveDocument.Range(Start:=oTable.Cell(2, 2).Range.Start, _
     End:=oTable.Cell(oTable.Rows.Count, 3).Range.End)
  oRange.Select
  Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
End Sub
```

(Ein zusammenhängender Zellbereich in einer Tabelle wird markiert und nachformatiert.)

Werte in Zellen formatieren

▲ Zurück zum Anfang

Im folgenden finden Sie ein Beispiel, wie sich Zellen in einer Tabelle sequentiell abarbeiten lassen. Sobald die Routine einen numerischen Wert in einer Tabelle entdeckt, wird dieser Wert als Geldbetrag formatiert.

```
Sub WerteInSpalteFormatieren()
  Dim oTable As Table, Zelle As Cell
  Set oTable = ActiveDocument.Tables(1)
  For Each Zelle In oTable.Range.Cells
    Betrag = Left(Zelle.Range.Text, Len(Zelle.Range.Text) - 2)
    If IsNumeric(Betrag) Then
      Betrag = Format(Betrag, "###'###'##0.00")
      ofs = InStr(Betrag, "'")
      While ofs = 1 'Strip leading Quotes
        Betrag = Mid(Betrag, 2)
        ofs = InStr(Betrag, "'")
      Wend
      Zelle.Range.Text = "DM " & Betrag
      Zelle.Range.ParagraphFormat.Alignment = wdAlignParagraphRight
    End If
  Next
End Sub
```

(Arbeitet den Inhalt von Tabellenzellen sequentiell ab und wandelt numerische Werte in Geldbeträge um.)

Letzte Tabellenzeile jeder Seite auffinden

▲ Zurück zum Anfang

Mitunter muss die letzte Zeile auf jeder Seite einer Tabelle, welche sich über mehrere Seiten erstreckt, ermittelt werden, um z.B. ein Zwischentotal beizufügen. Das folgende Beispiel zeigt die Mimik um dies zu realisieren. Die Mimik ist allerdings nicht sehr schnell.

```
Sub LetzeZeileAufSeiteFinden()
  savEnv = Options.Pagination
  Options.Pagination = False
  ActiveDocument.Repaginate
  Dim oTable As Table, Zeile As Row
  Set oTable = ActiveDocument.Tables(1)
  For Each Zeile In oTable.Rows
    If Zeile.IsLast Then
      Zeile.Cells(1).Range.Text = "Das ist die allerletzte Zeile."
    Else
      If Not Zeile.Range.Information(wdActiveEndPageNumber) = _
         Zeile.Next.Range.Information(wdActiveEndPageNumber) Then
            Zeile.Next.Cells(1).Range.Text = "Das ist die letzte Zeile."
      End If
    End If
  Next
  Options.Pagination = savEnv
End Sub
```

(Diese Routine findet jeweils sie letzte Zeile auf einer Seite.)

Formel in eine Zelle der letzen Tabellenzeile einfügen

▲ Zurück zum Anfang

Im Folgenden wird in der letzen Zeile (1. Spalte) die Formel *Summe(Über)* **Word 97** bzw. *Sum(Über)* **Word2000** eingefügt. Zur Sicherheit werden auch die Felder in der Tabelle aktualisiert.

```
Private Const Ausdruck = "=SUM(Über)"

Sub FormelInLetzeTabellenSpalte1()
  Dim oTable As Table, Zelle As Cell
  Set oTable = ActiveDocument.Tables(1)
  oTable.Cell(oTable.Rows.Count, 1).Formula Ausdruck
  oTable.Range.Fields.Update
End Sub
```

(In der letzten Tabellenzeile (1. Spalte) wird. eine Formel eingefügt.)

Hier noch eine elegantere (implizite) Lösung dieses Problems. Damit diese Mimik funktioniert, müssen in der Spalte allerdings schon Zahlen eingetragen sein.

```
Sub FormelInLetzeTabellenSpalte2()
  Dim oTable As Table, Zelle As Cell
  Set oTable = ActiveDocument.Tables(1)
  oTable.Cell(oTable.Rows.Count, 1).AutoSum
  oTable.Range.Fields.Update
End Sub
```

(In der letzten Tabellenzeile (1. Spalte) wird. die Formal *Summe Über* eingefügt.)

Verschachtelte Tabellen in Word2000 verarbeiten

▲ Zurück zum Anfang

Mit Word2000 kann man nun wie in HTML auch verschachtelte Tabellen erstellen. Eine Zelle kann nun also statt Text oder Grafik eine andere Tabelle beinhalten. Dies lässt sich dann für viele Ebenen weiterziehen.

Im folgenden finden Sie eine allgemeingültige Routine, um Zellen von verschachtelten Tabellen sequentiell abzuarbeiten. Dabei werden zuerst die Zellen der innersten Tabelle abgearbeitet. Dieses Vorgehen wird dann weiter gezogen bist zur der äussersten Tabelle.

Erstellen Sie eine (mehrfach) verschachtelte Tabelle, Kopieren Sie den Code in ein neues Modul und testen Sie das Makro.

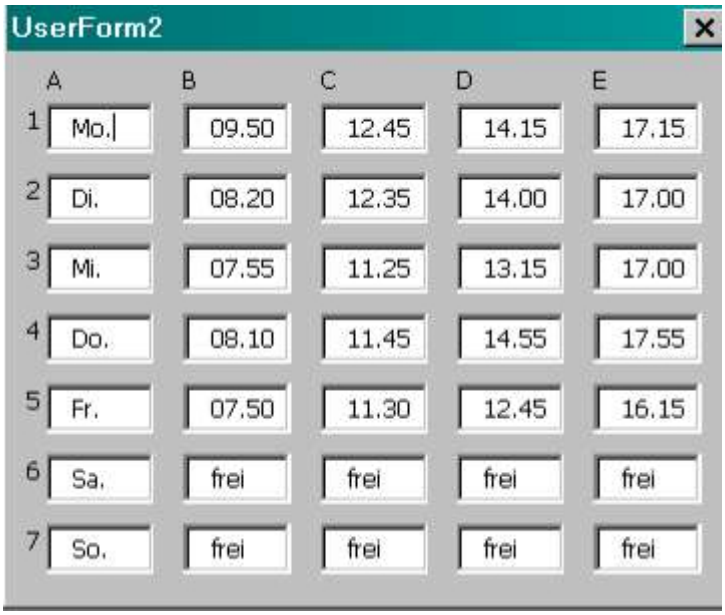
```
Private Zähler As Long

Sub Test()
  Zähler = 0
  RekursiveTabelleVerarbeiten ActiveDocument.Tables(1)
End Sub

Private Sub RekursiveTabelleVerarbeiten(oTable As Table)
  Dim oCell As Cell
  For Each oCell In oTable.Range.Cells
    If Not oCell.Tables.Count = 0 Then
      RekursiveTabelleVerarbeiten oCell.Tables(1)
    Else
      Zähler = Zähler + 1
      oCell.Range.Text = Zähler
    End If
  Next
End Sub
```

(Die Zellen einer verschachtelten Tabelle in Word2000 werden sequentiell verarbeitet, wobei die innere Tabelle Priorität hat.)

Word-Tabelle auf einer UserForm abbilden

▲ Zurück zum Anfang

Folgendes Beispiel bildet eine Tabelle, welches die erste Tabelle im Word-Dokument darstellt, auf einer UserForm ab.



Implementierung der Lösung

Um die Lösung zu testen, verfahren Sie wie folgt:

1. Erstellen Sie ein neues Word-Dokument und fügen sie eine Tabelle ein. Grundsätzlich kann diese Tabelle eine beliebige Anzahl von Zeilen und Spalten aufweisen. Das Programm ermittelt die Grösse der Tabelle dynamisch. Da die UserForm aber nicht beliebig gross werden sollte, ist die Grösse im Code bewusst limitiert (12 Zeilen bzw. 6 Spalten). Fühlen Sie sich aber frei, diese beiden Konstanten im Kopf des Prozesses mit dem Namen **DynamischeSchaltflächenTabelleAufruf** nach Ihren Bedürfnissen nach oben anzupassen. Die Namen der Konstanten lauten **MaxZeilen** bzw. **MaxSpalten**. Bedenken Sie aber dabei, dass die Anzahl der *Controls*, welche eine UserForm beherbergen kann nach oben limitiert ist.
2. Füllen Sie die Zellen der Tabelle mit etwas Text, damit das Ergebnis später auch sichtbar wird.
3. Wechseln Sie in den VBA-Editor und fügen Sie ein neues Module ein.
4. Kopieren Sie den Prozess mit dem Namen **DynamischeSchaltflächenTabelleAufruf** in dieses Modul. Beachten Sie, dass diese Routine davon aus geht, dass die UserForm den Namen *UserForm1* trägt. Allenfalls muss dieser Name von Ihnen noch angepasst werden.
5. Fügen Sie eine neue UserForm ein. Diese UserForm bleibt leer, das die Textfelder dynamisch, also mit Code erstellt werden.
6. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt ***Code anzeigen***.
7. Löschen Sie die zwei Code-Zeilen, welche Word automatisch generiert hat und kopieren Sie den unten stehenden Code, welcher für die UserForm bestimmt ist, in das nun leere Fenster.
8. Sie sind nun schon bereit, um die Sache zu testen.

VBA-Code

Aufruf der UserForm

Folgender Code tätigt die nötigen Vorarbeiten und ruft dann die UserForm auf.

```
Private Const MaxZeilen = 12
Private Const MaxSpalten = 6

Sub DynamischeSchaltflächenTabelleAufruf()
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Im Dokument befindet sich keine Tabelle."
    End
  End If
  Dim Tabelle As Table
  Set Tabelle = ActiveDocument.Tables(1)
  AnzZeilen = Tabelle.Rows.Count
  If AnzZeilen > MaxZeilen Then AnzZeilen = MaxZeilen
  AnzSpalten = Tabelle.Columns.Count
  If AnzSpalten > MaxSpalten Then AnzSpalten = MaxSpalten
  ReDim x(AnzZeilen - 1, AnzSpalten - 1) As String
  For i = 0 To AnzZeilen - 1
    For j = 0 To AnzSpalten - 1
      tmp = Tabelle.Cell(i + 1, j + 1).Range.Text
      x(i, j) = Left(tmp, Len(tmp) - 2)
    Next j
  Next i
  UserForm1.DynamischeSchaltflächenTabelleUserForm x()
End Sub
```

(Die Tabelle wird in ein zweidimensionales Datenfeld geladen, anschliessend wird die UserForm angezeigt..)

Der Code für die UserForm

Folgender Code erstellt dynamisch die nötigen Textfelder. Kopieren Sie den Code in eine leere User-Form.

```
Private Lab As MSForms.Label
Private TB As MSForms.TextBox

Sub DynamischeSchaltflächenTabelleUserForm(x() As String)
  Dim TB As MSForms.TextBox
  Titel = 24
  Header = 12
  Rand = 6
  StartOben = 3
  StartLinks = 6
  Höhe = 15
  Breite = 32
  AbsX = 9
  AbsY = 6
  Me.Height = ((UBound(x, 1) + 1) * Höhe) + (UBound(x, 1) * AbsY) + _
```

```
    (2 * StartOben) + Titel + Header
  Me.Width = ((UBound(x, 2) + 1) * Breite) + (UBound(x, 2) * AbsX) + _
    (2 * (StartLinks + Rand))
  oben = StartOben
  links = StartLinks + Rand
  For i = 0 To UBound(x, 2)
    Set Lab = Me.Controls.Add("Forms.Label.1", , True)
    With Lab
      .Top = oben
      .Left = links
      .Height = 9
      .Width = Breite
      .Caption = Chr(i + 65)
      links = links + Breite + AbsX
    End With
  Next i
  oben = StartOben + Header
  For i = 0 To UBound(x, 1)
  Set Lab = Me.Controls.Add("Forms.Label.1", , True)
  With Lab
    .Top = oben
    .Left = StartLinks
    .Height = 9
    .Width = 12
    .Caption = i + 1
    .TextAlign = fmTextAlignLeft
    oben = oben + Höhe + AbsY
  End With
  Next i
  oben = StartOben + Header
  For i = 0 To UBound(x, 1)
    links = StartLinks + Rand
    For j = 0 To UBound(x, 2)
      Set TB = Me.Controls.Add("Forms.TextBox.1", "TB" & Chr(j + 64) & i + 1, True)
      With TB
        .Top = oben
        .Left = links
        .Height = Höhe
        .Width = Breite
        .Text = x(i, j)
        links = links + Breite + AbsX
      End With
    Next j
    oben = oben + Höhe + AbsY
  Next i
  Me.Show
End Sub
```

(Die nötigen Textfelder werden dynamisch erstellt. Grösse der Textfelder und die Abstände zwischen den Feldern können Sie im Kopf des Codes allenfalls noch anpassen.)

Weitere VBA-Splitter im Zusammenhang mit Tabellen

▲ Zurück zum Anfang

Folgende Links zeigen aus sehr kompetenter Hand, weitere VBA-Splitter rund um Tabellen auf:

Microsoft Word MVP FAQ Site

- ► Doppelte Zeilen in einer Tabelle löschen
- ► Leere Zeilen in einer Tabelle löschen
- ► Feststellen, ob eine Tabellenzelle leer ist
- ► Alle Tabellenzeilen löschen, welche eine vorgegebene Zeichenfolge enthalten
- ► Index der Tabelle feststellen, in welcher sich die Einfügemarke befindet

Ein Excel-Tabellenblatt in einer Word-Tabelle abbilden

▲ Zurück zum Anfang

Im Folgenden finden Sie eine Routine, welche eine Excel-Tabelle in eine Word-Tabelle umwandelt (als Kopie). Die Tabelle wird am Ort der Einfügemarke in das Word-Dokument eingefügt.

Dank dem Einsatz von DAO (*Data Access Object*) kann diese Aufgabe sehr effizient gelöst werden, die Excel-Mappe muss dabei nämlich nicht mit einer trägen Office-Anwendung geöffnet werden.

Die Kennzeichnung, welche Daten übertragen werden sollen, geschieht in dem Sie einen benannten Bereich in der Excel-Tabelle definieren. Im Folgenden wird Schritt für Schritt erklärt, wie Sie konkret vorgehen sollten.

1. Implementierung des Beispieles
2. Öffnen Sie die Excel-Mappe mit den Daten, welche in einer Word-Tabelle abgebildet werden sollen.
3. Markieren Sie auf den Tabellenblatt den Bereich der Tabelle, welcher abgebildet werden soll.
4. Vom Hauptmenü in Excel wählen Sie ***Einfügen - Name - Definieren...***.
5. Vergeben Sie nun dem Bereich einen aussagekräftigen Namen. Speichern Sie dann die Excel-Mappe. Sie können die Excel-Anwendung jetzt schliessen.
6. Wechseln sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
7. Kopieren Sie den nachfolgenden Code in ein neues Modul.
8. Passen Sie nun im Kopf des Codes die Konstanten nach Ihren Gegebenheiten an. Die Konstante **Pfad** soll Pfad und Name Ihrer Excel-Arbeitsmappe tragen; und die Konstante **Bereich** soll den Bereichsnamen tragen, welchen Sie dem fraglichen Bereich in Excel vergeben haben.
9. Immer noch im VBA-Editor klicken Sie nun im Hauptmenü auf ***Extras - Verwiese...***.
10. Suchen Sie in der Liste nach einem Eintrag **Microsoft DAO** und aktivieren Sie diesen Verweis. Falls Sie mehrere Einträge bezüglich DAO vorfinden, aktivieren Sie die höchste (sprich neuste) Version
11. Speichern Sie nun das Werk und rufen Sie das Makro auf.

VBA-Code

```
Private Lab As MSForms.Label
Private Const Pfad = "C:\Eigene Dateien\Kantone der Schweiz.xls"
Private Const Bereich = "Kantone"

Sub ExcelTabelleInWordTabelle()
  Dim db As DAO.Database
  Dim tblObj As DAO.TableDef
  Dim rs As DAO.Recordset
  Set db = OpenDatabase(Pfad, False, True, "Excel 8.0;")
  DoEvents
  Q = Chr(34)
  Set rs = db.OpenRecordset("Select * FROM `" & Bereich & "`")
  For i = 0 To rs.Fields.Count - 1
    strT = strT & rs.Fields(i).SourceField & Chr(1)
  Next
  While Not rs.EOF
    strT = Left(strT, Len(strT) - 1)
    strT = strT & Chr(13)
    For j = 0 To rs.Fields.Count - 1
      strT = strT & rs.Fields(j).Value & Chr(1)
    Next j
    rs.MoveNext
  Wend
  strT = Left(strT, Len(strT) - 1)
  Selection.InsertBefore strT
  Selection.ConvertToTable Separator:=Chr(1), _
    Format:=wdTableFormatColorful2, AutoFit:=True
  Selection.Collapse Direction:=wdCollapseStart
  Set rs = Nothing
  Set tblObj = Nothing
  Set db = Nothing
End Sub
```

(Aus einem *Benannten Bereich* in einer Excel-Tabelle wird unter Einsatz von *DAO* eine Word-Tabelle. Die Excel-Arbeitsmappe braucht dabei nicht mit einer trägen Office-Anwendung geöffnet werden.)

Eine Access-Tabelle in einer Word-Tabelle abbilden

▲ Zurück zum Anfang

Einleitung

Im Folgenden finden Sie eine Routine, welche eine Access-Tabelle in eine Word-Tabelle umwandelt (als Kopie). Die Tabelle wird am Ort der Einfügemarke in das Word-Dokument eingefügt.

Dank dem Einsatz von ADO (*ActiveX Data Object*) kann diese Aufgabe sehr effizient gelöst werden, die Access-MDB muss dabei nämlich nicht mit einer trägen Office-Anwendung geöffnet werden.

Das Beispiel bezieht sich auf eine MS Access Datenbank, welche Sie als Beispiel auf Ihrer Office 97 CD finden können. Die Datei hat den Namen **Nordwind.mdb**. In der Office2000-Ausgabe ist diese mdb auch vorhanden, allerdings unter einem etwas anderen Namen.

Diese Datenbank besteht aus diversen Tabellen. Für unsere Beispiele habe ich die Tabelle mit dem Namen **Bestellungen** ausgewählt. Diese Tabelle wiederum beinhaltet ein kleines überschaubares Set an Datenfelder und etwa 80 Datensätze.

Es kommt eine **SQL-Anweisung** zur Anwendung, um bekannt zu geben, welche Felder übertragen werden sollen. SQL ist das Akronym für *Structered Query Language*. Es handelt sich dabei glücklicherweise um eine herstellerunabhängige Methode, welche weite Verbreitung findet. Sie werden darüber - z.B. im WEB - sehr viel Literatur finden, um die Sache noch zu verfeinern. Sie können mit einer einzigen SQL-Anweisung eigentlich alles notwendige mitgeben, namentlich:

- Den Namen der **Tabelle**, welche Sie ansprechen möchten.
- Die **Feldinhalte** dieser Tabelle, welche Sie zurück haben möchten.
- Einfache oder auch anspruchsvolle **Abfrage-Optionen**.
- Letztlich können Sie auch noch gleich die **Sortierreihenfolge** der Datensätze vorschreiben..

Ergebnis

Das Ergebnis des Codes ist eine schön formatierte Tabelle, welche alle Datensätze in der Tabelle **Bestellungen** anzeigt. Aufgelistet werden folgende Feldinhalte: *Lieferdatum*, *Empfänger*, *Ort* und *Bestimmungsland*. Sie können diese Auswahl selbstverständlich Ihren Bedürfnissen anpassen.

Implementierung des Beispieles

Um das Beispiel einzurichten, gehen Sie bitte wie folgt vor:

1. Beschaffen Sie sich die Access-Datenbank **Nordwind**, welche Sie als Beispiel auf Ihrer Festplatte oder auf Ihrer Office-CD vorfinden.
2. Wechseln Sie mit der Tastenkombination [**Alt + F11**] in die VBA-Umgebung.
3. Dort angekommen, wählen Sie vom Hauptmenü den Dialog ***Extras - Verweise...***.
4. In der erscheinenden Liste aktivieren Sie den Verweis auf die Objektbibliothek mit dem Namen **Microsoft ActiveX Data Objects x.x Object Library**. Sollte die Liste mehrere Versionen von ADO anzeigen, wählen Sie die neuere, sprich höhere Version.
5. Kopieren Sie den folgenden VBA-Code in ein neues Modul, passen Sie allenfalls Pfad und Name der mdb im Kopf des Codes noch an und testen Sie das Makro.

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\Nordwind.mdb"

Sub WordTabelleAusAccessDatenGenerieren()
  Dim oWord As Word.Application
  Dim oDoc As Word.Document
  Dim oRange As Word.Range
  Dim oConn As ADODB.Connection
  Dim oRS As ADODB.Recordset
  Set oRange = Selection.Range
  Set oConn = CreateObject("ADODB.Connection")
  oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
     Pfad & ";Persist Security Info=False"
  Set oRS = oConn.Execute( _
    "SELECT Lieferdatum, Empfänger, Ort, Bestimmungsland FROM Bestellungen")
  sTemp = oRS.GetString(adClipString, -1, vbTab)
  sTemp = "Lieferdatum" & vbTab & "Empfänger" & vbTab & _
     "Ort" & vbTab & "Bestimmungsland" & vbCrLf & sTemp 'Tabellenkopf
  oRange.Text = sTemp
  oRange.ConvertToTable Separator:=vbTab, Format:=wdTableFormatColorful2
  Set oWord = Nothing
  Set oDoc = Nothing
  Set oConn = Nothing
  Set oRS = Nothing
End Sub
```

(Aus einer Access-Tabelle wird eine wohlformatierte Word-Tabelle. Die Tabelle wird am Ort der Einfügemarke in das Word-Dokument eingefügt.)

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Wie kann ich in einem VBA-Programm (Makro) feststellen, ob sich die Einfügemarke in einer Tabelle befindet?
► Tabelle auf einer UserForm abbilden
► Mit "DAO" Daten aus einem Excel-Tabellenblatt ziehen, ohne die Arbeitsmappe zu öffnen
► Mittels "ADO" auf Access zugreifen

Microsoft Word MVP FAQ Site

► Deleting duplicate rows in a table
► Deleting all empty rows in a table using a macro
► Detect whether a table cell is empty
► Delete all rows of a table that contain a particular text string in the first column
► Determine the index number of the current paragraph, table, section ...

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Tabellen.htm
René Probst, Januar 2001

Änderungen

- 19. Januar 2001 - Zeilen und Spalten hinzufügen
- 20. Januar 2001 - Arbeiten mit dem Range-Objekt von Tabellen
- 25. Februar 2001 - Tabellenspalten verarbeiten

VBA-technischer Umgang mit Textmarken

◄ Zurück zum Hauptindex

Diese Seite

▼ Einleitung
▼ Arten von Textmarken und ihr Verhalten
▼ Anspringen von Textmarken versus Arbeiten mit der Range-Eigenschaft von Textmarken
▼ Feststellen, ob eine bestimmte Textmarke im Dokument vorhanden ist
▼ Textstelle zwischen zwei Textmarken finden bzw. markieren
▼ Wie Textmarken die Arbeit erleichtern können
▼ Textmarkentext mit benutzerspezifischen Werte füllen
▼ Tool für das schnelle Navigieren zwischen Textmarken im Word-Dokument
▼ Links zu diesem Thema

Einleitung

▲ Zurück zum Anfang

Word-VBA bietet sehr gute Schnittstellen, um mit Textmarken zu arbeiten. Dabei ist es eigentlich nicht nötig zu der fraglichen Textmarke zu springen oder sogar in eine andere Ansicht zu wechseln (falls sich die Textmarke z.B. im Kopfzeilenbereich befindet). Textmarken lassen sich nämlich über ihre *Range*-Eigenschaft (Bereich) ansprechen.

Word-VBA bietet überdies auch eine komfortable Abfragemöglichkeit, um festzustellen, ob eine Textmarke mit einem bestimmten Name im Dokument überhaupt vorhanden ist.

Arten von Textmarken und ihr Verhalten

▲ Zurück zum Anfang

Was viele zu wenig beachten, Word kennt eigentlich zwei Arten von Textmarken, nämlich offene und geschlossene Textmarken. Um dies zu erkennen, machen wir folgenden Test:

Offene Textmarken

! Damit Textmarken überhaupt am Bildschirm angezeigt werden, sollten Sie zuerst die Einstellung ***Extras - Optionen - Ansicht - Anzeigen - Textmarken*** überprüfen.

Setzen Sie nun die Einfügemarke im Dokument an die gewünschte Stelle, und wählen Sie **ohne dass etwas im Dokument markiert wäre**, den Dialog ***Einfügen - Textmarke...*** und vergeben Sie der Marke einen Namen. Das Resultat ist eine **offene Textmarke** und wird wie folgt angezeigt: [. Mit diesem Vorgehen, haben Sie letztlich eine **Position** im Dokument definiert.

Weisen Sie mittels VBA bei dieser Art von Textmarke einen Text zu, also z.B. *Hallo Welt* und Sie lassen diesen Code zehn mal ablaufen, dann steht dort 10x Hallo Welt. Der Text nach der Textmarke wird dabei immer nach recht geschoben um Platz zu schaffen. Die Textmarke selbst ist dabei nicht betroffen, sie bleibt erhalten an der selben Stelle. Die Textmarke repräsentiert ja eine Position und eine Position kann man physisch nicht löschen.

Geschlossene Textmarken (Textmarkenfelder)

Als nächstes markieren Sie nun einen kurzen Text z.B. ein Wort im Dokument. und wählen dann wieder den Dialog ***Einfügen - Textmarke...*** dann vergeben Sie der Marke einen Namen. Das Resultat ist eine **geschlossene Textmarke** und wird wie folgt angezeigt: `[ ]`. Sie haben letztlich einen **Bereich** definiert. In der Word-Hilfe heisst so ein Ding übrigens **Textmarkenfeld**.

Weisen Sie mittels VBA bei dieser Art der Textmarke einen Text zu, also z.B. *Hallo Welt* und Sie lassen diesen Code zum zweiten Mal ablaufen, dann wird es nicht mehr klappen, viel mehr werden Sie eine Fehlermeldung erhalten. Der Bereich wurde nämlich beim ersten Durchlauf überschrieben und zwar inklusive Textmarke.

Dieses Verhalten ist oft praxisgerecht, ja sogar erwünscht; allerdings nicht immer. Das nächste Kapitel zeigt ein dynamisches System auf, also eine Textmarke, deren Textmarkentext ich beliebig oft manipulieren kann. Dabei wird der bestehende Text mit dem neuen Text überschrieben, die Textmarke bleibt dabei aber faktisch erhalten. Eigentlich wird mit einem Trick gearbeitet: die Textmarke wird einfach am Schluss wieder bei der ursprünglichen Stelle gesetzt. Deshalb habe ich diese Mimik ***Recycling Textmarke*** getauft. Den Trick habe ich übrigens zum ersten mal bei **Cindy Meister** gesehen.

Recycling Textmarke ex VBA

Damit das folgende Experiment gelingt muss eine **geschlossene Textmarke** (Textmarkenfeld) mit dem Namen **MeineMarke** definiert werden. Was innerhalb der geschlossenen Textmarke steht, ist völlig irrelevant. Wichtig ist der Name der Marke, welcher mit der Syntax im Code korrespondieren muss.

```
Sub RecyclingTextmarke()
  TMName = "MeineMarke"
  If ActiveDocument.Bookmarks.Exists(TMName) Then
    Dim TMRange As Range
    Set TMRange = ActiveDocument.Bookmarks(TMName).Range
    TMRange.Text = Time
    ActiveDocument.Bookmarks.Add Name:=TMName, Range:=TMRange
  End If
End Sub
```

(Das *Recycling* einer geschlossenen Textmarke.)

Anspringen von Textmarken versus Arbeiten mit der *Range*-Eigenschaft von Textmarken

▲ Zurück zum Anfang

Erfahrungsgemäss wählen Einsteiger beim Verarbeiten von Textmarken einen vielleicht naheliegenden aber letztlich doch umständlichen Weg. Sie springen nämlich zur fraglichen Textmarke mit der *GoTo*-Methode und tun dann dort etwas.

Dies geht solange einigermassen gut, bis Textmarken nicht in einem anderen Dokumententeil (*StoryRange*), also z.B. in der Kopfzeile angesprungen werden sollen. Dann versagt diese Methode oder sie wird umständlich, da zuerst die Ansicht gewechselt werden muss, also von der Normalansicht in die Kopf-/Fusszeilenansicht.

Es ist nun aber gar nicht nötig, eine Textmarke anzuspringen, da auch dieses Objekt über die *Range*-Eigenschaft verfügt. Den Bereich, welche Sie bearbeiten möchten, muss ja nicht mit dem identisch sein, wo sich die Einfügemarke gerade befindet. Das Arbeiten mit Ranges ist übrigens wesentlich performander als das herumhüpfen im Dokument und erst noch weniger störanfällig.

Sollten Sie Word fernsteuern, also von einer anderen Anwendung auf Word mittels *Office Automation* (*GetObject*- und *CreateObject*-Methode zugreifen, wird das Herumhüpfen im Dokument noch um eine Dimension problematischer, vor allem dann wenn die Word-Instanz gar nicht sichtbar sondern im Hintergrund ist.

Die nachfolgenden Beispiel zeigen einen Vergleich zwischen der *Selection.GoTo*- und der *Range*-Methode, wobei die *Range*-Methode wohl in allen belangen besser abschneidet.

```
Sub TextmarkeBearbeiten1()
  Selection.GoTo What:=wdGoToBookmark, Name:="Ort"
  Selection.TypeText Text:="Basel"
End Sub
```

(Oft wird zu einer Textmarke gesprungen und dann Text eingegeben. Diese Vorgehensweise ist naheliegend, aber nicht optimal.)

```
Sub TextmarkeBearbeiten2()
  ActiveDocument.Bookmarks("Ort").Range.Text = "Basel"
End Sub
```

(Bewirkt das selbe, wie das obige Beispiel, allerdings ohne im Dokument herum zu hüpfen. Diese Methode ist der ersteren klar vorzuziehen.)

Angenommen, die Textmarke befindet sich nicht im Hauptteil des Dokumentes, also z.B. in der Kopfzeile, dann wird es mit der *Selection.GoTo*-Methode ziemlich umständlich und auch fehleranfällig. Zuerst muss nämlich in die Kopf-/Fusszeilenansicht gewechselt werden, damit der *GoTo*-Befehl erfolgreich ist. Diese Problematik umgehen Sie geschickt, wenn Sie das *Range*-Objekt einsetzten, wie dies oben bereits gezeigt wurde, Der Code ist also universell anwendbar; sie können beliebige Dokumententeile ansteuern.

Bedenken Sie, dass wenn Sie den Code wiederholt ablaufen lassen, sich der Text bei der Textmarke kumuliert (bei einer offenen Textmarke), oder dass die Textmarke nach dem ersten Durchlauf gelöscht wird (Textmarkenfeld). Um diese Sache in den Griff zu bekommen, lesen Sie das vorhergehende Kapitel *Arten von Textmarken und ihr Verhalten*.

Feststellen, ob eine bestimmte Textmarke im Dokument vorhanden ist

▲ Zurück zum Anfang

Sie sollten es sich zur Gewohnheit machen, zu testen, ob eine bestimmte Textmarke, welche Sie ansprechen möchten überhaupt im Dokument vorhanden ist. Falls Sie dies nicht tun und die Textmarke ist nicht vorhanden, dann bricht der Code nämlich mit einer Fehlermeldung ab. Wir erweitern also unser Beispiel wie folgt:

```
Sub TextmarkeBearbeiten3()
  If ActiveDocument.Bookmarks.Exists("Ort") Then
    ActiveDocument.Bookmarks("Ort").Range.Text = "Basel"
  End If
End Sub
```

(Nachdem abgefragt wurde, ob die fragliche Textmarke existiert, wird bei der Textmarke Text eingefügt.)

Diese Mimik lässt sich übrigens auch bei konventionellen **Formulartextfeldern** einsetzen, verfügen diese doch auch über einen Textmarkenname.

Im folgenden Beispiel wird zuerst geprüft, ob das Formularfeld mit dem Namen **Text1** existiert. Ist dies der Fall, wird dieses Formularfeld mit einem Wert bestückt.

```
Sub TestenObFormulartextfeldExistiert()
  If ActiveDocument.Bookmarks.Exists("Text1") Then
    ActiveDocument.FormFields("Text1").Result = "Hallo Welt"
  End If
End Sub
```

(Auch konventionelle Formulartextfelder verfügen über einen Textmarkennamen. Deshalb lässt sich die beschriebene Mimik auch auf Formularfelder ausweiten.)

Textstelle zwischen zwei Textmarken finden bzw. markieren

▲ Zurück zum Anfang

Die Frage sei hier erlaubt, wieso Sie mit zwei Textmarken arbeiten an Stelle eines **Textmarkenfeldes**, wie dies schon im Kapitel Arten von Textmarken und ihr Verhalten abgehandelt wurde. Sie würden im letzteren Fall mit einem einzigen Textmarkenfeld, welches ja einen Bereich umfasst, auskommen.

Bei einer offenen Textmarke dagegen bestimmen Sie keinen Bereich sondern einen Punkt im Dokument. Somit sind zwei Textmarken von Nöten, nämlich für das Anzeigen des Startpunktes wie auch für den Endpunkt.

Trotzdem, wenn Sie auf zwei Textmarken beharren, dann können Sie Ihr Vorhaben wie es in den folgenden beiden Beispielen demonstriert wird, erreichen.

Die Beispiele gehen davon aus, dass es sowohl eine Textmarke mit dem Namen **MeinAnfang** wie auch eine Textmarke mit dem Namen **MeinEnde** im Dokument gibt.

```
Sub TextZwischenZweiTextmarkenMarkieren()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  If oDoc.Bookmarks.Exists("MeinAnfang") And _
     oDoc.Bookmarks.Exists("MeinEnde") Then
    oDoc.Range(Start:=oDoc.Bookmarks("MeinAnfang").Range.Start, _
       End:=oDoc.Bookmarks("MeinEnde").Range.Start).Select
  End If
End Sub
```

(Markiert den Text zwischen zwei Textmarken; im Beispiel heissen diese Textmarken *MeinAnfang* bzw. *MeinEnde*.)

```
Sub TextZwischenZweiTextmarkenRangeBilden()
  Dim oDoc As Document, oRange As Range
  Set oDoc = ActiveDocument
  If oDoc.Bookmarks.Exists("MeinAnfang") And _
     oDoc.Bookmarks.Exists("MeinEnde") Then
    Set oRange = _
       oDoc.Range(Start:=oDoc.Bookmarks("MeinAnfang").Range.Start, _
          End:=oDoc.Bookmarks("MeinEnde").Range.Start)
    oRange.Select
  End If
End Sub
```

(Bildet einen *Range*, welcher den Bereich zwischen zwei Textmarken umfasst.)

Wie Textmarken die Arbeit erleichtern können

▲ Zurück zum Anfang

Ein Word-Dokument besitzt nicht die selbe starre Struktur, wie wir es z.B. von einer Excel-Tabelle kennen. Die Hauptstärke von MS Word liegt in der Verarbeitung von **Fliesstext** und da treten scheinbare Strukturelemente wie Zeilen oder Seiten in den Hintergrund. Sollten in einem Word-Dokument Zeilen oder Seiten das Ziel Ihres Unterfangens sein, dann begeben Sie sich auf Glatteis. Sie sollten einen solchen Ansatz nach Möglichkeit vermeiden. Textmarken können Ihnen dabei helfen.

Lösen Sie sich auch von der Vorstellung, welche Sie z.B. von Excel mit sich bringen. Wenn Sie eine neue Excel-Arbeitsmappe erstellen, erscheint dort ein Tabellenblatt mit einem vorgegebenen Raster, viele Zellen, welche zu einer Spalte und gleichzeitig zu einer Zeile gehören.

Sie können nun hingehen und in Excel die Zelle *C12* abfragen oder etwas rein schreiben. Die Abfrage wird zwar bei einer leeren Tabelle einen leere Zeichenfolge zurückgeben aber die Zelle ist da und adressierbar.

Anders bei MS Word. Erstellen Sie ein neues Word-Dokument, dann ist da wirklich gar nichts vorhanden; ausser einer einzelnen einsamen Absatzmarke. So sieht es nicht nur für den Benutzer aus, es ist Word-intern auch wirklich so.

Im Gegensatz zu der Zelle *C12* auf einem Excel-Tabellenblatt ist die «Ortsangabe» Paragraph 3, Zeile 4, Position 12 also rein hypothetisch, will heissen vielleicht gibt es ja dieses Element gar noch nicht.

Anders wenn Sie an dieser Stelle eine Textmarke einfügen, dann muss diese Stelle ja wenigsten existieren. Eine Textmarke ist also nicht hypothetisch, sie ist entweder vorhanden, oder nicht. Im letzteren Fall müssen Sie dafür sorgen, dass sie wirklich vorhanden ist.

Diese Textmarke lässt sich nun problemlos ansteuern, wie dies in den vorhergehenden Kapiteln bereits demonstriert wurde. Dies ist in der Praxis wesentlich einfacher und vor allem sicherer als zum Beispiel einen bestimmten Absatz - oder noch exotischer in einem Fliesstext - eine bestimmte Zeile anzusteuern.

Nehmen wir mal an, Sie verwenden in Ihrem Word-Dokument Textfelder und möchten diese nun ansteuern um ein bestimmten Text dort einzugeben. Diese Aufgabenstellung ist so zwar lösbar aber nicht ganz einfach zu erreichen, nicht zuletzt deshalb, weil Word die Textfelder ziemlich zufällig benennt und es sich erst noch um Unterobjekte einer sogenannten *Shape* handelt.

Wesentlich einfacher fahren Sie, wenn Sie eine Textmarke mit einem eindeutigen Namen in das Textfeld setzten. Nun können Sie diese Textmarke wie hier beschrieben ansteuern. Dass sich die Textmarke in einem Textfeld befindet und der Name des Textfeldes, ist dabei gar nicht relevant.

Textmarkentext mit benutzerspezifischen Werte füllen

▲ Zurück zum Anfang

Mitunter taucht der Wunsch auf, bestehende Textmarken welche sich ursprünglich in einer Dokumentvorlage befinden dann aber vererbt werden, wenn ein neues Dokumentes basierend auf dieser Vorlage erstellt wird, automatisiert mit Text zu bestücken. Der Text, welcher eingefügt wird, soll dabei vom Benutzer abhängig sein.

Genau mit dieser Aufgabenstellung befasst sich ein anderes Skript in dieser Reihe. Sie finden dazu den Link am Ende dieser Abhandlung.

Tool für das schnelle Navigieren zwischen Textmarken im Word-Dokument

▲ Zurück zum Anfang

Haben Sie sich auch schon geärgert, dass das Anspringen von Textmarken in einem Dokument über den integrierten Word-Dialog *Einfügen - Textmarken* ziemlich umständlich ist?

Wenn Sie drei ;-) der oben gestellten Fragen mit *Ja* beantworten können, dann sollten Sie sich das Skript aus dieser Reihe ansehen, welches zwei Tools vorstellt, um Textmarken besser in den Griff zu bekommen. Sie finden den Link dazu finden Sie am Ende dieser Abhandlung.

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Textmarkentext mit benutzerspezifischen Standardwerten belegen
► Wie lassen sich Textmarken in einem Word-Dokument bequem ansteuern?
► Arbeiten mit Range-Objekten in MS Word

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/TM.htm

René Probst, Januar 2001

Änderungen

- --- nil ---

Wie kann ich in einem VBA-Programm (Makro) feststellen, auf welcher Seite sich die Einfügemarke befindet?

Wie kann ich in einem VBA-Programm (Makro) feststellen, ob sich die Einfügemarke in einer Tabelle befindet?

◄ Zurück zum Hauptindex

Frage

> 1. Ich sollte in meiner VBA-Routine feststellen können, auf welcher Seite sich die Einfügemarke befindet.
>
> 2. Für ein Gelingen meiner VBA-Routine muss ich wissen, ob sich die Einfügemarke in einer Tabelle befindet oder nicht.

Diese Seite

- ▼ Die Information-Eigenschaft
- ▼ Feststellen, wie viele Seiten das Dokument umfasst
- ▼ Feststellen, auf welcher Seite sich die Einfügemarke befindet
- ▼ Feststellen, auf welcher Seite sich ein bestimmtest Objekt befindet
- ▼ Feststellen, wie viel Platz auf der Seite noch frei ist
- ▼ Text verkleinern/vergrössern, bis er genau auf eine Zeile bzw. genau auf eine Seite passt
- ▼ Feststellen, ob sich die Einfügemarke in einer Tabelle befindet
- ▼ Feststellen, in welcher Tabellen-Zelle sich die Einfügemarke befindet
- ▼ Feststellen, ob sich die Einfügemarke in einem Textfeld befindet

Die *Information*-Eigenschaft

▲ Zurück zum Anfang

Die **Information**-Eigenschaft liefert nützliche Informationen was die Position der Einfügemarke, Markierung oder eines Objektes betrifft. Die nützlichsten dieser Informationen sind in der folgenden Tabelle zusammen getragen.

Befehl	Bedeutung
`SeitenNr = _` `  Selection.Information(wdActiveEndPageNumber)`	Liefert die Nummer der **Seite** auf welcher sich die Einfügemarke befindet. Sind mehrere Seiten markiert, ist das Ende der Markierung massgebend.
`KorrigierteSeitenNr = _` `  Selection.Information(wdActiveEndAdjustedPageNumber)`	Ähnlich wie oben, nur liefert dieser Ausdruck die **angepasste** (benutzerdefinierte) **Seitennummer**. Der Benutzer kann bekanntlich auf Seite 5 definieren, dass die Zählung der Seiten nun bei 100 beginnen soll oder er kann auch römische Zahlen etc. vergeben. Es handelt sich dabei um die Information, welche der Benutzer auf der Statuszeile unter **Seite** vorfindet.
`AbschnittNr = _` `  Selection.Information(wdActiveEndSectionNumber)`	Liefert die Nummer des **Abschnittes** auf welcher sich die Einfügemarke befindet. Sind mehrere Abschnitte markiert, ist das Ende der Markierung massgebend. Es handelt sich dabei um die Information, welche der Benutzer auf der Statuszeile unter **Ab** vorfindet.
`InTabelle = _` `  Selection.Information(wdWithInTable)`	Gibt den Wert True zurück, falls sich die **Einfügemarke in einer Tabelle befindet**.
`SpaltenNr = _` `  Selection.Information(wdEndOfRangeColumnNumber)`	Liefert die Nummer der **Tabellenspalte** in der sich die Einfügemarke befindet. Ist der Zeiger nicht in einer Tabelle, wird der Wert **-1** zurückgegeben.
`ZeilenNr = _` `  Selection.Information(wdEndOfRangeRowNumber)`	Liefert die Nummer der **Tabellenzeile**, in der sich die Einfügemarke befindet. Ist der Zeiger nicht in einer Tabelle, wird der Wert **-1** zurückgegeben.
`Spalte = _` `  Selection.Information(wdFirstCharacterColumnNumber)`	Liefert die **Spaltennummer** in der sich die Einfügemarke befindet. Sind mehrere Zeichen markiert, ist die Position des ersten Zeichens massgebend. Es handelt sich dabei um die Information, welche der Benutzer auf der Statuszeile unter **Sp** vorfindet.
`Zeile = _` `  Selection.Information(wdFirstCharacterLineNumber)`	Liefert die **Zeilennummer** in der sich die Einfügemarke befindet. Sind mehrere Zeilen markiert, ist die erste markierte Zeile massgebend. Es handelt sich dabei um die Information, welche der Benutzer auf der Statuszeile unter **Ze** vorfindet.

`Bei = _`
`  Format(Selection.Information(wdVerticalPositionRelativeToPage) _`
`  * 0.035, "#.0")`
	Liefert den Abstand vom oberen Papierrand in Zentimeters.
	Es handelt sich dabei um die Information, welche der Benutzer auf der Statuszeile unter **Bei** vorfindet.

Feststellen, wie viele Seiten das Dokument umfasst

▲ Zurück zum Anfang

Quasi ausser Konkurrenz hier der VBA-Code, welcher ermittelt, wie viele Seiten das Dokument umfasst. Es gibt dazu verschiedene Möglichkeiten. Die sicherste Variante ist jene, welche mit der *ComputeStatistics*-Methode arbeitet. Bedenken Sie, dass MS Word um diese Seitenzahl zu ermitteln den Vorgang des **Seitenumbruches** auslösen und zu Ende führen muss. Dies ist mit der folgenden Mimik, implizit gewährleistet.

```
Sub AnzahlSeitenErmitteln()
  MsgBox ActiveDocument.ComputeStatistics(wdStatisticPages)
End Sub
```

(Die Anzahl der Seiten im Dokument wird ermittelt.)

Feststellen, auf welcher Seite sich die Einfügemarke befindet

▲ Zurück zum Anfang

Im folgenden werden verschiedene Mimiken aufgezeigt, um zu ermitteln, auf welcher Seite sich die Einfügemarke befindet. Es beginnt mit der einfachsten Mimik, welche aber auch gewisse Gefahren in sich birgt, nämlich die falsche Seite zurück zu bekommen.

Bedenken Sie, dass MS Word um zu ermitteln, auf welcher Seite sich die Einfügemarke befindet, streng genommen den Vorgang des Seitenumbruches durchführen muss. Bedenken Sie weiter, dass MS Word es ermöglicht, die Umgebung so einzurichten, dass dieser Seitenumbruch im Hintergrund stattfindet. In diesem Fall stellt das Umbrechen der Seiten ein asynchronen Prozess dar.

Da Sie beim Programmieren ohne Abfrage nicht wissen können, wie der Benutzer die Word-Oberfläche in diesem Bereich konfiguriert hat (*Extras - Optionen - Allgemein - Seitenumbruch im Hintergrund)* sollten Sie hier die sicherste der im Folgenden diskutierten Varianten wählen.

```
MsgBox Selection.Information(wdActiveEndPageNumber)
```

(Ermittelt die Seite, in welcher sich die Einfügemarke befindet.)

Im folgenden Beispiel wird ebenfalls ermittelt, auf welcher Seite sich die Einfügmarke befindet. Zur Sicherheit wird aber zuerst der Vorgang des Seitenumbruches ausgelöst, um zu verhindern, dass die Seite falsch ermittelt wird.

```
Sub SeiteErmitteln2()
  ActiveDocument.Repaginate
  MsgBox Selection.Information(wdActiveEndPageNumber)
End Sub
```

(Bevor die Seite ermittelt wird, sollte sicher gestellt werden, dass die Seiten im Dokument bereits richtig umgebrochen sind.)

Da es sich beim Vorgang des Seitenumbruches aber um einen asynchronen Vorgang handeln kann, sollten wir dafür sorgen, dass dieser Vorgang auch beendet ist, bevor wir quasi was messen. Die nächsten zwei Beispiele sollen diesen Gedanken in die Tat umsetzen. Bezüglich der Sicherheit sind für mich die folgenden Beispiele gleichwertig. Das erste Beispiel sorgt implizit, das zweite Beispiel explizit für sichere Bedingungen.

```
Sub SeiteErmitteln3()
  ActiveDocument.ComputeStatistics wdStatisticPages
  MsgBox Selection.Information(wdActiveEndPageNumber)
End Sub
```

(Eine elegante Lösung, um der möglichen Asynchronität aus dem Wege zu gehen.)

```
Sub SeiteErmitteln4()
  savEnv = Options.Pagination
  Options.Pagination = False
  ActiveDocument.Repaginate
  MsgBox Selection.Information(wdActiveEndPageNumber)
  Options.Pagination = savEnv
End Sub
```

(Die klassische und explizite Lösung um der möglichen Asynchronität aus dem Wege zu gehen.)

Bedenken Sie, dass alle bisher vorgestellten Lösungen, das Ende der Markierung, bzw. wenn nichts markiert ist die Lage der Einfügemarke bestimmen. Möchten Sie erfahren, auf welcher Seite die Markierung beginnt, dann verwenden Sie folgenden Code.

```
Sub AufWelcherSeiteBeginntDieMarkierung()
  ActiveDocument.ComputeStatistics wdStatisticPages
  MsgBox ActiveDocument.Range(Selection.Range.Start, _
     Selection.Range.Start).Information(wdActiveEndPageNumber)
End Sub
```

(Hier wird ermittelt, auf welcher Seite die Markierung *beginnt*.)

Feststellen, auf welcher Seite sich ein bestimmtest Objekt befindet

▲ Zurück zum Anfang

Die ***Information*-Eigenschaft** lässt sich auch problemlos auf ***Range*-Objekte** anwenden. Um festzustellen, auf welcher Seite sich die erste Tabelle im Dokument befindet, verwenden Sie folgenden Code:

```
Sub FeststelleAufWelcherSeiteDieErsteTabelleImDokumentEndet()
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument beinhaltet keine Tabelle."
    Exit Sub
  End If
  ActiveDocument.ComputeStatistics wdStatisticPages
  MsgBox ActiveDocument.Tables(1).Range.Information(wdActiveEndPageNumber)
End Sub
```

(Hier wird die Seitennummer der ersten Tabelle im Dokument ermittelt. Ausgewiesen wird die Seite mit welcher die Tabelle *endet*.)

Dieser Code liefert also jene Seitenzahl, mit welcher die Tabelle endet. Wenn Sie erfahren möchten, auf welcher Seite die erste Tabelle im Dokument **beginnt**, verwenden Sie folgenden Code:

```
Sub FeststelleAufWelcherSeiteDieErsteTabelleImDokumentBeginnt()
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument beinhaltet keine Tabelle."
    Exit Sub
  End If
  ActiveDocument.ComputeStatistics wdStatisticPages
  With ActiveDocument.Tables(1)
    MsgBox ActiveDocument.Range(.Range.Start, _
       .Range.Start).Information(wdActiveEndPageNumber)
  End With
End Sub
```

(Hier wird die Seitennummer der ersten Tabelle im Dokument ermittelt. Ausgewiesen wird die Seite auf welcher die Tabelle *beginnt*.)

Feststellen, wie viel Platz auf der Seite noch frei ist

▲ Zurück zum Anfang

Bei einigen Aufgabenstellungen ist es wichtig, zu erfahren, wie viel Platz auf der Seite noch bleibt Sie möchten ja vielleicht erfahren, ob ein weiteres Element noch ganz auf der Seite Platz findet, oder ob Sie besser eine neue Seite beginnen sollten.

Im folgenden wird ausgewiesen, wie viel Platz bis zum Ende der Seite bleibt. Ausgangspunkt ist dabei die Position der Einfügemarke. Der verbleibende Platz wird im Beispiel in Zentimeter ausgedrückt.

```
Sub WievielPlatzBleibtAufDerSeite1()
  'a = Seitenhöhe des fraglichen Abschnittes
  'b = Position der Einfügemarke ab oberen Blattrand
  'c = Unterer Blattrand des fraglichen Abschnittes
  'd = Noch verfügbarer Platz
  If ActiveWindow.View.SplitSpecial = wdPaneNone Then
    ActiveWindow.ActivePane.View.Type = wdPrintView
  Else
    ActiveWindow.View.Type = wdPrintView
  End If
  Dim Abschnitt As Section
  Set Abschnitt = ActiveDocument.Sections(Selection.Information(wdActiveEndSectionNumber))
  ActiveDocument.ComputeStatistics wdStatisticPages
  a = Abschnitt.PageSetup.PageHeight
  b = Selection.Information(wdVerticalPositionRelativeToPage)
  c = Abschnitt.PageSetup.BottomMargin
  d = PointsToCentimeters(a - b - c)
  MsgBox d & " cm"
End Sub
```

(Weisst aus, wie viel Platz auf der Seite bleibt. Ausgangspunkt ist dabei die Einfügemarke.)

Im folgenden wird ausgewiesen, wie viel Platz hinter der ersten Tabelle des Dokumentes auf der Seite noch frei ist.

```
Sub WievielPlatzBleibtAufDerSeite2()
  'a = Seitenhöhe des fraglichen Abschnittes
  'b = Position der Einfügemarke ab oberen Blattrand
  'c = Unterer Blattrand des fraglichen Abschnittes
  'd = Noch verfügbarer Platz
  If ActiveWindow.View.SplitSpecial = wdPaneNone Then
    ActiveWindow.ActivePane.View.Type = wdPrintView
  Else
    ActiveWindow.View.Type = wdPrintView
  End If
  Dim Abschnitt As Section, oRange As Range
  Set oRange = ActiveDocument.Tables(1).Range
  oRange.SetRange Start:=oRange.End, End:=oRange.End
  Set Abschnitt =
ActiveDocument.Sections(oRange.Information(wdActiveEndSectionNumber))
  ActiveDocument.ComputeStatistics wdStatisticPages
  a = Abschnitt.PageSetup.PageHeight
  b = oRange.Information(wdVerticalPositionRelativeToPage)
  c = Abschnitt.PageSetup.BottomMargin
  d = PointsToCentimeters(a - b - c)
  MsgBox d & " cm"
End Sub
```

(Weisst aus, wie viel Platz nach der ersten Tabelle im Hauptteil des Dokumentes auf der Seite noch bleibet.)

Text verkleinern/vergrössern, bis er genau auf eine Zeile bzw. genau auf eine Seite passt

▲ Zurück zum Anfang

Folgendes Beispiel vergrössert den Text in der Zeile in welcher sich die Einfügemarke befindet solange, bis dieser Text genau in diese, eine Zeile passt: Dabei wird der Schriftgrad variiert.

```
Sub TextInZeileEinpassen()
  Dim A As Range, B As Range
  Selection.HomeKey
  Set A = Selection.Bookmarks("\Line").Range
  If Len(A.Text) < 2 Then
    MsgBox "Die Zeile enthält keinen Text."
    Exit Sub
  End If
  Set B = Selection.Bookmarks("\Line").Range
  A.SetRange Start:=A.Start, End:=A.Start
  B.SetRange Start:=B.End - 1, End:=B.End - 1
  While A.Information(wdFirstCharacterLineNumber) = _
        B.Information(wdFirstCharacterLineNumber)
    Selection.Bookmarks("\Line").Range.Font.Size = Selection.Font.Size + Step
    If Selection.Font.Size + Step > 1638 Then Exit Sub
  Wend
  Selection.Bookmarks("\Line").Range.Font.Size = Selection.Font.Size - Step * 2
End Sub
```

(Die Schrift eines markierten Textes wird stufenweise so lange verkleinert, bis der Text genau in eine Zeile passt)

Folgendes Beispiel verkleinert bzw. vergrössert den Hauptteil des Dokumentes so lange, bis dieser genau auf eine Seite passt. Dafür wird sowohl der Schriftgrad, wie auch der Durchschuss (Abstand zwischen den Zeilen) variiert.

```
Sub TextAufSeiteEinpassen()
  savEnv = Options.Pagination
  Options.Pagination = False
  ActiveDocument.Content.ParagraphFormat.LineSpacingRule = wdLineSpaceAtLeast
  Do
    Grad = Selection.Font.Size
    ActiveDocument.Content.Font.Size = Grad + Step
    ActiveDocument.Content.ParagraphFormat.LineSpacing = (Grad + Step) * Durchschuss
   Loop While ActiveDocument.ComputeStatistics(wdStatisticPages) = 1 And _
     Selection.Font.Size + Step < 1638
  Do
    Grad = Selection.Font.Size
    ActiveDocument.Content.Font.Size = Grad - Step
    ActiveDocument.Content.ParagraphFormat.LineSpacing = (Grad + Step) * Durchschuss
  Loop While ActiveDocument.ComputeStatistics(wdStatisticPages) > 1 And _
     Selection.Font.Size - Step > 1
  Options.Pagination = savEnv
End Sub
```

(Die Schrift des Hauptteils im Dokument wird stufenweise so lange vergrössert bzw. verkleinert, bis der Text genau auf eine Seite passt)

Feststellen, ob sich die Einfügemarke in einer Tabelle befindet

▲ Zurück zum Anfang

Gewisse Abläufe gelingen nicht, wenn sich die Einfügemarke beim Start eines Makros in einer Tabelle befindet. Andere Routinen wiederum setzen gerade voraus, dass sich der Zeiger in eine Tabelle befindet. Dank der *Information-Eigenschaft* ist es einfach festzustellen, ob sich die Einfügemarke in einer Tabelle befindet.

Folgende Routine bricht ab, nachdem einen Hinweis an den Benutzer ausgegeben wurde, falls sich der Zeiger in einer Tabelle befindet:

```
Sub FeststellenObSichDieEinfügemarkeInEinerTabelleBefindet()
  If Selection.Information(wdWithInTable) Then
    Text = "Sie können diese Routine nicht durchführen, weil "
    Text = Text & "sich die Einfügemarke in einer Tabelle befindet."
    MsgBox Text, vbExclamation
  End If
  ' Ihr Code
  ' Ihr Code
  ' Ihr Code
End Sub
```

(Diese Routine wird nicht durchgeführt, falls sich die Einfügemarke in einer Tabelle befindet.)

Feststellen, in welcher Tabellen-Zelle sich die Einfügemarke befindet

▲ Zurück zum Anfang

Das folgende Code-Beispiel zeigt dem Benutzer an, in welcher Zelle sich die Einfügemarke befindet. Sind mehrere Zellen markiert, wird die Start. und Endzelle ausgewiesen. Die Anzeige erfolgt auf die selbe Weise, wie Sie es sich von Excel gewohnt sind. Steht der Zeiger z.B. in der zweiten Zeile und in der vierten Spalte, wird **D4** angezeigt. Diese Routine ist vor allem dann praktisch, wenn Sie Formeln in eine Tabelle einfügen wollen. In einer Formel müssen Sie bekanntlich die Zellen auf diese Art und Weise adressieren.

Diese Routine verrichtet hier immer wieder wertvolle Dienste:

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Priavte Const Titel = "Tabellen Position"

Sub TabellenPositionAnzeigen()
   If Selection.Information(wdWithInTable) = False Then
      Markierung = 0
      If Selection.Type = 1 Then
        Meldung = "Der Zeiger befindet sich nicht in einer Tabelle!"
      Else
        Meldung = "Die Markierung stellt keinen gültigen Tabellenbereich dar!"
      End If
   Else
      If Selection.Type = 1 Then
         Position = TabellenPosition
         Markierung = 1
         tmp = Position
         Meldung = "Die aktive Zelle heisst: " & tmp & "."
      Else
         Bereich = TabellenBereich
         Markierung = 2
         tmp = Bereich
         Meldung = "Der markierte Tabellenbereich heisst: " & tmp & "."
      End If
   End If
   MsgBox Meldung, vbInformation, Titel
End Sub

Private Function TabellenPosition() As String
  TabellenPosition = _
     AlphaNum(Selection.Information(wdStartOfRangeColumnNumber)) & _
        Selection.Information(wdStartOfRangeRowNumber)
End Function

Private Function TabellenBereich() As String
   TabellenBereich = _
      AlphaNum(Selection.Information(wdStartOfRangeColumnNumber)) & _
         Selection.Information(wdStartOfRangeRowNumber) & " - " & _
            AlphaNum(Selection.Information(wdEndOfRangeColumnNumber)) & _
               Selection.Information(wdEndOfRangeRowNumber)
End Function

Private Function AlphaNum(Num As Long) As String
  Z = 0
  While Num > 26
    Z = Z + 1
    Num = Num - 26
  Wend
  AlphaNum = Chr(Num + 64)
  If Z > 0 Then AlphaNum = Chr(Z + 64) & AlphaNum
End Function
```

(Dieser Code teilt dem Benutzer mit, in welcher Zelle sich die Einfügemarke befindet Sind mehrere Zellen markiert, wird die Start- und Endzelle ausgewiesen.)

Neue Tabelle in das Dokument einbringen

Die folgenden beiden Code-Beispiele sind funktionell identisch - unterscheiden sich aber durch die Art, wie das Problem angegangen wird. Der erste Code verwendet die *Hüpf-hüpf-Methode* (Navigation mit dem **Selection-Objekt**), während der zweite Code mit einem **Range-Objekt** arbeitet, um die richtige stelle zu erreichen.

Das Ziel ist es, eine neue Tabelle in das Dokument einzubringen. Zuerst wird festgestellt, ob sich die Einfügemarke in einer Tabelle befindet, wenn ja wird die Stelle direkt unterhalb dieser Tabelle avisiert, wenn nein wird keine neue Stelle gesucht sondern direkt beim Zeiger eine weitere Tabelle eingefügt. Würde die Abfrage, ob sich bei der Einfügemarke bereits eine Tabelle befindet unterbleiben, würde **Word 97** die Routine mit einem Fehler abbrechen. Mit **Word2000** dagegen, würde eine verschachtelte Tabelle entstehen.

```
Sub TabelleEinfügen1()
  With Selection
    If .Information(wdWithInTable) = True Then
      .MoveDown Count:=.Tables(1).Rows.Count - _
         .Information(wdEndOfRangeRowNumber) + 2
    End If
    ActiveDocument.Tables.Add Range:=.Range, NumRows:=2, NumColumns:=2
  End With
End Sub
```

(Eine Tabelle wird in das Word-Dokument eingebracht. Sollte sich an der Stelle, wo die Einfügemarke ist, bereits eine Tabelle befinden, wird ausgerechnet, um wie viele Zeilen der Zeiger nach unten hüpfen muss.)

```
Sub TabelleEinfügen2()
  Dim oRange As Range
  Set oRange = Selection.Range
  If Selection.Information(wdWithInTable) = True Then
    oRange.SetRange Start:=Selection.Tables(1).Range.End + 1, _
                    End:=Selection.Tables(1).Range.End + 1
  End If
  ActiveDocument.Tables.Add Range:=oRange, NumRows:=2, NumColumns:=2
End Sub
```

(Verfolgt das selbe Ziel, wie das obige Beispiel. Diesmal wird aber mit einem Range-Objekt gearbeitet. Diese Lösung ist der obigen vorzuziehen.)

Feststellen, ob sich die Einfügemarke in einem Textfeld befindet

▲ Zurück zum Anfang

Mit der Information-Eigenschaft können Sie nicht feststellen, ob sich die Einfügemarke in einem Textfeld befindet. Benutzen Sie dazu folgende Mimik:

```
Sub FeststellenObSichDieEinfügemarkeInEinemTextfeldBefindet()
  If Selection.StoryType = wdTextFrameStory Then
    MsgBox "Die Einfügemarke befindet sich in einem Textfeld."
  Else
    MsgBox "Die Einfügemarke befindet sich nicht in einem Textfeld."
  End If
End Sub
```

(Stellt fest, ob sich die Einfügmarke in einem Textfeld befindet.)

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/SelInfo.htm

René Probst, Oktober 1999

Änderungen

- 4. Mai 2000 - Beispiele mit Range-Objekten
- 4. Mai 2000 - Beispiele mit Tabellen
- 14. November 2000 - Problematik mit Seitenumbruch
- 6. Februar 2001 - Neu strukturiert und erweitert

Textgebundene und Textungebundene Grafiken und Objekte («InlineShapes» und «Shapes»)

◄ Zurück zum Hauptindex

Diese Seite

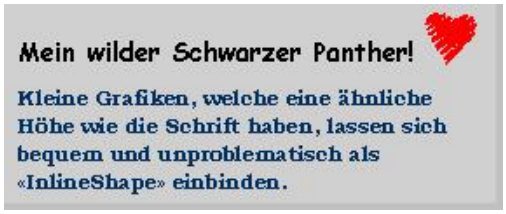
- Einleitung
- Woran erkenne ich, ob ich eine «Shape» oder eine «InlineShape» vor mir habe?
- Wieso macht MS Word die Unterscheidung zwischen «Shapes» und «InlineShapes»?
- Wie kann ich beeinflussen, ob eine «Shape» oder eine «InlineShape» entsteht?
- Wie kann ich Word dazu bewegen den Standardwert, ob beim Einfügen eine «Shape» oder eine «InlineShape» entsteht ändern?
- Arten von «Shapes» und «InlineShapes»
- «Shape» in «InlineShape» umwandeln und v.v.
- Charakteristika von «Shapes» und «InlineShapes»
- Beschriftungen und Abbildungsverzeichnis
- Eine «Shape» oder eine «InlineShape» verschieben
- Objekte an einer bestimmten Stelle auf der Seite festnageln
- Objekte, welche sich auf jeder Seite wiederholen in das Dokument einbringen
- Ein Wasserzeichen in das Dokument einbringen
- VBA-technischer Umgang mit «Shapes» und «InlineShapes»
- Tabelle der «StoryRanges» (Dokumententeile) in MS Word
- Links zu diesem Thema

Einleitung

▲ Zurück zum Anfang

Was viele Anwender und Anwenderinnen zu wenig beachten, ist die Tatsache, dass MS Word bei der Gestaltung von Word-Dokumenten, welche Grafiken und Objekte enthalten, zwei ganz verschiedene Möglichkeiten bietet, diese Objekt einzubinden:

- Objekte können **textbezogen** eingebunden werden und liegen in diesem Fall immer auf der selben Ebene, wie der Text selbst. Eine Überlappung ist in diesem Falle nicht möglich. Greifen Sie zu dieser Methode, wenn Sie ein Objekt in ein Textfeld oder in eine Tabellenzelle einfügen möchten Auch ohne zu diesen Strukturmitteln zu greifen, mag es bei sehr kleinen Objekten, Sinn machen, diese auf diese Art und Weise einzubetten. Im folgenden wird diese Art von eingebundenen Objekten ***InlineShape*** genannt.



- Objekte können aber auch **seitenbezogen** eingebunden werden und liegen in diesem Fall **nicht** auf der selben Ebene, wie der Text selbst. Eine Überlappung mit dem Text ist in diesem Falle als Spezialfall möglich. Meist richtet man es aber so ein, dass der Text um das Objekt fliesst. Die Art und Wiese, wie dieser Textfluss geschehen soll, kann von Ihnen beeinflusst werden. Objekte, welche in dieser Art und Weise eingebunden werden, können nur in der **Seitenlayoutansicht** betrachtet und bearbeitet werden. So wie Sie diese Methode zum einbinden wählen schaltet Word automatisch in die Seitenlayoutansicht, sollte dies nicht bereits geschehen sein. Im folgenden wird diese Art von eingebundenen Objekte ***Shape*** genannt.

Versinnbildlichen Sie den Unterschied von einer Shape und einer InlineShape wie folgt:

Bei der **InlineShape** stellen Sie sich eine Perlenkette vor, welche aus vielen weissen Glasperlen und einige roten Glasperlen besteht. Die weissen Perlen symbolisieren den Text oder genauer gesagt die Zeichen (Buchstaben). Die roten Perlen dagegen stehen für die InlineShapes. Die Perlen stehen in Reih' und Glied sie können sich nicht überlagern. Sie können allerdings hin gehen und eine der roten Perlen an einer anderen Stelle einreihen. Damit rücken die Perlen am alten Ort zusammen und am neuen Ort auseinander.

Eine InlineShape reiht sich wie die roten Perlen in der Grafik in den Text ein. Sie önnen diese Reihenfolge ändern, aus der Reihe tanzen kann eine InlineShape allerdings nicht. Ein Textumfluss ist damit nicht möglich.

Als Sinnbild für die **Shape**, stellen Sie sich ein Blatt Papier vor, welches mit Sand bestreut ist und ein Dreieck, welches Sie aus einem anderen Bogen Papier ausgeschnitten haben. Das Papier stellt unsere Seite in Word dar, das Dreieck steht für die Shape. Der Sand wiederum symbolisiert den Text auf Ihrer Seite. Legen Sie nun dieses farbige Dreieck auf das Papier. Es können nun verschiedene Konstellationen entstehen und all diese Situationen werden von Word unterstützt. Sie können dies beeinflussen, wenn Sie den Textfluss rund um die Shape neu einstellen. Somit kann das Dreieck auf dem Sand liegen, es kann unter dem Sand liegen oder der Sand kann weichen und die Kontur des Dreiecks freilegen. Um einen weiteren englischen Begriff zu bemühen: *A Shape is floating*.

Das selbe Bild – einmal als «Shape»...

Woran erkenne ich, ob ich eine «*Shape*» oder eine «*InlineShape*» vor mir habe?

▲ Zurück zum Anfang

Nun Sie erkennen dies vor allem daran, wie Word diese Objekte nach dem Markieren kenntlich macht. Insbesondere die Handhaben, an denen Sie das Objekt skalieren können sehen unterschiedlich aus. Bei einer **Shape** erkennen Sie **kleine transparente Quadrate**, während die **InlineShape** mit einem **schwarzen Rahmen** daherkommt.

... und als «InlineShape».

Achten Sie auf die Unterschiedliche Markierung.

«Shapes» und «InlineShape» sehen gleich aus, zeigen aber ein unterschiedliches Verhalten.

Wieso macht MS Word die Unterscheidung zwischen «*Shapes*» und «*InlineShapes*»?

▲ Zurück zum Anfang

Die Frage könnte genau so gut lauten: «Wieso bittet eine gute Eisenwarenhandlung sowohl Nägel wie auch Schrauben an?» Richtig ist, dass für manche Arbeiten, sowohl das eine wie auch das andere verwendet werden kann. Das sind aber eher die Ausnahmen. In der Praxis ist es meistens so, dass das eine klar dem anderen vorzuziehen ist. Auch Methoden, welche Text ganz anders darstellen als Word - ich denke dabei z.B. an HTML - unterscheiden zwischen der seitenlayout- und der textbezogenen Darstellung.

☛ Eine Regel ist gewiss: sollte Ihre Absicht sein, die Grafik (oder das Objekt) in ein Textfeld oder in eine Tabellenzelle einzufügen, dann verwenden Sie eine InlineShape, Sollten Sie diese Regel missachten, landet Ihre Grafik irgendwo nur nicht in der Tabelle oder im Textfeld.

! Beachten Sie weiter, dass nicht alle Arten von Objekten sowohl als Shapes wie auch als InlineShapes fungieren können. Genaueres darüber, entnehmen Sie bitte der Tabelle, welche Sie etwas weiter unten in diesem Skript finden.

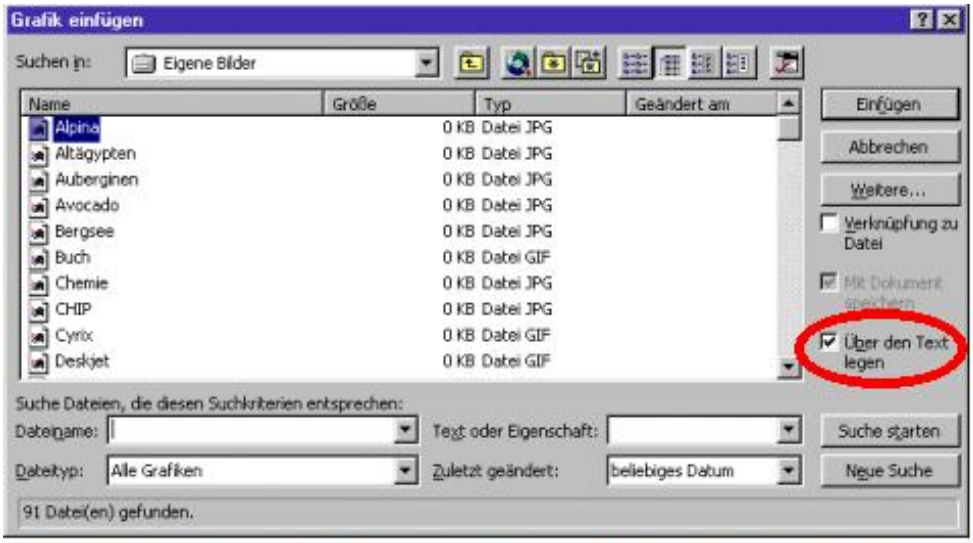
Wie kann ich beeinflussen, ob eine «*Shape*» oder eine «*InlineShape*» entsteht?

▲ Zurück zum Anfang

Das Vorgehen hängt davon ab, welche Word-Version bei Ihnen im Einsatz steht.

Word 97

Ausschlaggebend in Word 97, ob letztlich eine Shape oder eine InlineShape entsteht, ist die Option **Über den Text legen**, welche Sie als Kontrollkästchen in den entsprechenden Word-Dialogen vorfinden. Ist diese Option **eingeschaltet** entsteht eine **Shape**, ist sie dagegen **ausgeschaltet**, entsteht eine **InlineShape**.

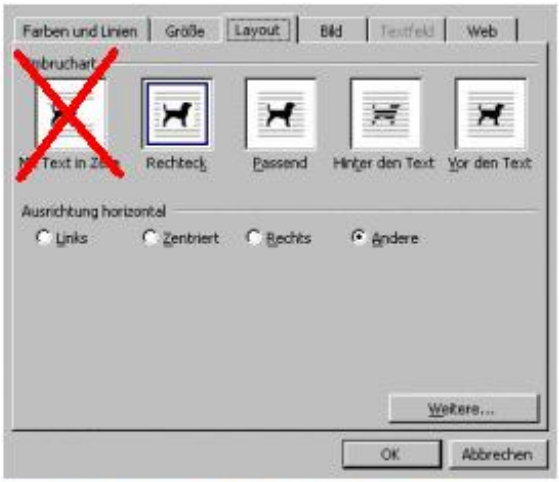


Das Attribut «*Über den Text legen*» entschied in Word 97 letztlich noch darüber, ob eine freie (schwebende) Grafik oder eine Grafik, welche sich wie ein Zeichen verhält, entsteht.

Word2000

Standardmässig entsteht eine InlineShape. Möchten Sie diese InlineShape in eine Shape umwandeln, gehen Sie wie folgt vor:

1. Führen Sie einen Rechtsklick auf das Objekt aus.
2. Wählen Sie im nun erscheinenden Kontextmenü den Befehl **Grafik formatieren..., Objekt formatieren...** oder worum es sich letztlich auch immer handelt.
3. Im nun erscheinenden Dialog wechseln Sie in den Reiter **Layout**.
4. Wählen Sie im Abschnitt *Umbruchart* im Katalog eine **andere** Umbruchart als *Mit Text in Zeile*.



Standardmässig fügt Word2000 eine Grafik als «InlineShape» ein. Um diese «InlineShape» in eine «Shape» umzuwandeln, muss die Umbruchart anders sein als «Mit Text in Zeile».

Wie kann ich Word dazu bewegen den Standardwert, ob beim Einfügen eine «Shape» oder eine «InlineShape» entsteht ändern?

▲ Zurück zum Anfang

Leider können Sie dies nicht wirklich ändern, bzw. es würde sehr viele Eingriffe (mittels VBA) bedeuten, da zahlreiche Word-Dialoge betroffen sind.

Es scheint so, dass sich weder Microsoft intern noch die Kunden von Microsoft auf **den** Standard einigen können, jedenfalls wird man mit jeder neuen Version von Word -- und sei es auch nur nachdem ein Service Release eingespielt wurde -- von neuem überrascht. Das ist selbstverständlich irritierend. Ich versuche aus dem Gedächtnis eine Chronologie aufzuzeigen, was über die Zeit mit diesem Häkchen ***Über den Text legen*** geschehen ist, welches ja letztlich darüber entscheidet, ob eine Shape oder eine InlineShape geboren wird:

- **Vor Word 97** war *«Über den Text legen»* standardmässig deaktiviert; es wurde also eine InlineShape eingefügt. **Mit Word 97** hat sich dies dann geändert. Heftige Proteste eines guten Teils der Word-Anwender war die Folge.
- Mit dem **Office 97 Service Release 1** kam eine positive Überraschung. Von nun an merkte sich Word die letzte Einstellung, welche der Benutzer angewendet hatte.
- Nach dem Einspielen des **Office 97 Service Release 2** musste der Anwender konsterniert feststellen, dass die Verbesserung, welche der Service Release 1 in dieser Sache gebracht hatte wieder zu Nichte war. Applaus, Microsoft!
- Word2000 Benutzer und Benutzerinnen werden feststellen, dass die Option *«Über den Text legen»* in dieser Form gar nicht mehr existiert. Die entsprechenden Dialoge habe ihr Aussehen stark verändert. Standardmässig wird eine InlineShape geboren, was für die meisten Word-Benutzer nochmals eine Umstellung in zweifacher Hinsicht bedeutet (der Dialog und der Standard hat geändert!). Wie Sie bei Bedarf diese InlineShape in eine Shape umwandeln können, wurde im vorhergehenden Kapitel bereits dargelegt.

Arten von «Shapes» und «InlineShapes»

▲ Zurück zum Anfang

Die nachfolgende Tabelle soll verdeutlichen, welche Objekte als Shapes bzw. als InlineShapes in ein Dokument eingebracht werden können.

Bitte beachten Sie als eine Grundregel, dass Objekte, welche selbst wieder als Gefässe für Text oder anderes fungieren können (darunter fallen insbesondere Textfelder und AutoFormen), nur als Shapes auftreten können.

Objekt	Shape	Inline-Shape
Grafiken (also z.B. JPG-Bilder etc.)	x	x
Textfelder	x	
AutoFormen	x	
Legenden (AutoFormen - Legende)	x	
Linien und Kurven	x	
Polygone (Rechtecke, Dreiecke, etc.)	x	
WordArt-Objekte	x	x
Steuerelemente (OLE Controls) , welche Sie über die Symbolleiste Steuerelement-Toolbox einbringen können	x	x
OLE Objekte also z.B. eine Excel-Tabelle, ein Excel-Diagramm, eine Formel aus dem Formeleditor, etc.	x	x
Etiketten	x	

«Shape» in «InlineShape» umwandeln und v.v.

▲ Zurück zum Anfang

Um eine Shape in eine InlineShape umzuwandeln verfahren Sie bitte wie folgt:

Mittels Word-Dialog

Word 97

1. Führen Sie einen Rechtsklick auf das Objekt aus.
2. Wählen Sie im nun erscheinenden Kontextmenü, je nach dem, um welche Art von Objekt es sich handelt entweder ***Grafik formatieren...*** bzw. ***Objekt formatieren...***.
3. Wechseln Sie nun in den Reiter **Position**.
4. **De**aktivieren Sie nun das Kontrollkästchen mit dem Namen **Über den Text legen**.

Um eine InlineShape in eine Shape umzuwandeln verfahren Sie genau gleich, wie oben beschrieben; diesmal aktivieren Sie aber das Kontrollkästchen mit dem Namen *Über den Text legen*.

Word2000

1. Führen Sie einen Rechtsklick auf das Objekt aus.
2. Wählen Sie im nun erscheinenden Kontextmenü, je nach dem, um welche Art von Objekt es sich handelt entweder ***Grafik formatieren...*** bzw. ***Objekt formatieren...***.
3. Wechseln Sie nun in den Reiter **Layout**.
4. Wünschen Sie eine **InlineShape**, dann aktivieren Sie im Katalog Umbruchart ***Mit Text in Zeile***, möchten Sie dagegen eine Shape wählen Sie irgend eine andere Umbruchart, als die erwähnte.

Mittels VBA

Eingangs sei erwähnt, dass sich der Unterschied zwischen Shape und InlineShape auch in VBA niederschlagen muss. Banal aber wichtig: Sprechen sie eine Shape immer als Shape an und eine InlineShape immer als InlineShape. Beachten Sie, dass sich Eigenschaften und Methoden für Shapes und InlineShapes in wesentlichen Punkten unterscheiden. Die Umwandlung erfolgt wie folgt:

```
ActiveDocument.Shapes(1).ConvertToInlineShape
ActiveDocument.InlineShapes(1).ConvertToShape
```

(Konvertierung einer Shape in eine InlineShape und v.v.)

Charakteristika von «Shapes» und «InlineShapes»

▲ Zurück zum Anfang

Die nachfolgende Tabelle gibt noch mal im Detail wieder, wie sich *InlineShapes* im Vergleich mit *Shapes* verhalten.

Eigenschaft/ Verhalten	Shape	InlineShape
Positionierung	Kann Millimeter- (bzw. Pixel-genau) geschehen. Dies geschieht unabhängig vom Text. Die Positionierung erfolgt im Bezug auf die Seite (Seitenrand).	Das Objekt liegt auf der selben Ebene, wie der Text. Man kann das Objekt vor oder nach einem anderen Zeichen positionieren. Es ist und bleibt aber an den Text gebunden.
Textfluss	In der Regel setzt sich eine Shape dem Text in den Weg. Je nach der Umbruchart, welcher der Benutzer wählt, wird das Objekt in der einten oder anderen Art vom Text umflossen. Auch eine Überlagerung der Shape mit dem Text ist möglich.	Da eine InlineShape, selbst Bestandteil des Textes ist, erübrigt sich die Frage nach dem Textfluss.
Verbergen	Um zu verhindern, dass Shapes ausgedruckt werden, können Sie folgende Option deaktivieren: Extras - Optionen - Drucken - Zeichnungsobjekte Damit wird der Druck aller Shapes unterdrückt. Es besteht keine Möglichkeit eine einzelne Shape selektiv zu verbergen. Die genannte Option wirkt sich immer auf alle Shapes aus.	Um zu verhindern, dass eine InlineShape ausgedruckt wird, markieren Sie die betreffende InlineShape und wählen den Dialog: Format - Zeichen - Verborgen **Word 97** bzw. Format - Zeichen - Ausgeblendet **Word2000**
Umwandlung	Eine Shape kann in eine InlineShape umgewandelt werden.	Eine InlineShape kann in eine Shape umgewandelt werden
Container	Gewisse Shapes (Textfelder und AutoFormen) können wiederum als Container für Text oder Objekte fungieren	Eine InlineShape kann nicht als Container fungieren.

Beschriftungen und Abbildungsverzeichnis

▲ Zurück zum Anfang

Beschriftungen und das Abbildungsverzeichnis haben zwar im technischen Sinne nichts mit Shapes oder InlineShapes zu tun, arbeitstechnisch gehören sie aber doch ziemlich eng zusammen. Dies hat mich dazu bewogen, auch hier auf die Tücken dieser Elemente hinzuweisen.

Ganz am Ende des Skriptes, finden Sie im übrigen einen Verweis zu einem anderen Skript, welches Ihnen erläutert, was zu tun ist, wenn bei der Erstellung des Abbildungsverzeichnisses keine Einträge gefunden werden, obwohl in der Tat solche vorhanden sind.

Aber beginnen wir von vorne.

Wenn Sie, nachdem Sie z.B. eine Grafik in das Dokument eingefügt haben, dieser Grafik über ***Einfügen - Beschriftung...*** eine Bildlegende zufügen möchten, dann sind genau drei Szenarien möglich:

1. Die Einfügemarke steht unter einer **InlineShape**.
2. Die Einfügemarke steht unter einer **Shape**. Die Shape ist dabei **nicht markiert**.
3. Die Einfügemarke steht unter einer **Shape**. Die Shape ist dabei markiert.

Bei den **Szenarien 1 + 2** sind keine besonderen Vorkehrungen nötig.. Auch die Erstellung eines Abbildungsverzeichnisses sollte kein Kopfzerbrechen bereiten. Sie werden feststellen. dass diese Beschriftungen automatisch in das Verzeichnis aufgenommen werden.

Sonderbar ist das Verhalten von Word, wenn das dritte Szenario angewandt wird. Wenn Sie entsprechend vorgehen, wird Word ohne Ihr hinzu tun die Beschriftung in ein Textfeld stellen. Dieses Textfeld wird dann halt von Word erstellt. Die Idee dahinter ist wohl, dass Sie die beiden Elemente - nämlich Grafik und Beschriftung - als nächstes gruppieren können und diese Gruppe nun frei verschieben können und zwar in einer Weise, dass dieses Paar nicht auseinander fällt. Da sowohl eine Grafik, welche als Shape eingebracht wird, wie ein Textfeld eine Shape darstellt, ist die Summe davon wieder eine Shape. Somit ist das machbar. Mit einer InlineShape jedenfalls, wäre dieses Spiel nicht möglich.

Diese Methode hat aber einen entscheidenden Nachteil. Das Problem ist, dass Textfelder auf einer anderen Ebene liegen, als der Hauptteil des Dokumentes. Dies hat zur Folge, dass wenn Sie ein Abbildungsverzeichnis erstellen, Beschriftungen in Textfeldern nicht gefunden werden. Sie fehlen dann einfach, oder wenn Ihr Dokument gar keine anderen Arten von Beschriftungen enthält, kann gar kein Abbildungsverzeichnis erstellt werden.

Sie können diese Textfelder in Positionsrahmen umwandeln, dann klappt das mit dem Abbildungsverzeichnis; oder aber Sie können ganz auf des dritte Szenario verzichten, den die Sache ist von Microsoft wirklich nicht gut abgestimmt.

Wie schon erwähnt, erfahren Sie mehr über diese Sache in einem Link, welchen Sie ganz am Schluss dieser Abhandlung finden.

Eine «Shape» oder eine «InlineShape» verschieben

▲ Zurück zum Anfang

Shapes

Eine Shape kann im Dokument frei bewegt werden und dies sogar **pixelgenau**. Ein Pixel entspricht übrigens einem zweiundsiebzigstel (1/72) eines Zolls (*Die spinnen dir Briten!)* - und das sind ca. 0.035 cm. Sie werden aber wahrscheinlich feststellen, dass wenn Sie das Objekt ziehen, sich dieses Objekt ruckartig und nicht kontinuierlich bewegt.

Standardmässig ist MS Word nämlich so eingestellt, das sich die Objekte an einem gedachten Raster ausrichten. Möchten Sie nun aber das Objekt stufenlos verschieben, dann halten Sie während dieses Vorganges einfach die Taste **[Alt]** gedrückt.

Alternativ können Sie dieses gedachte Raster auch enger machen. Dazu blenden Sie die Symbolleiste mit dem Namen **Zeichnen** ein. Auf dieser Leiste klicken Sie auf den Schaltknopf, welcher ebenfalls **Zeichnen** heisst, und wählen im nun geöffneten Menü den Befehl ***Gitternetz...***. Im folgenden Dialog können Sie dann die Rastereinstellungen verändern und das Raster bei Bedarf sogar sichtbar machen.

InlineShapes

Auch eine InlineShape kann verschoben werden. Allerdings verhalten sich Objekte, welche als InlineShape eingebracht wurden, wie Zeichen. Sie können also InlineShapes nur im Bezug auf den Textfluss verschieben und nicht in Bezug auf das Saitenlayout. Auf diese Tatsache wurde in diesem Skript schon mehrere Male hingewiesen.

Objekte an einer bestimmten Stelle auf der Seite festnageln

▲ Zurück zum Anfang

Immer wieder taucht die Frage auf, wie man ein Objekt (z.B. eine Grafik) an einer bestimmten Stelle des Dokumentes im Bezug auf das Seitenlayout festmacht, damit diese nicht mehr verrutscht. Der Benutzer soll diese Objekte also gar nicht mehr verschieben können. Das ist eigentlich in MS Word gar nicht möglich. Word handelt mit Fliesstext und Text fliesst und die Objekte fliessen eben mit.

Es gibt zwar sehr wohl für Shapes eine **Verankerung** (und nur für Shapes), damit erreichen Sie aber lediglich, dass das Objekt relativ zum Absatz an dem er seinen Anker ausgeworfen positioniert ist. Das will heissen, dass dieser Absatz den Bezugspunkt abgibt, wenn Sie die Koordinaten des Objektes festlegen.

Möchten Sie erreichen, dass sich das Objekt mit bewegt, wenn Sie einen Absatz verschieben, welcher einen Anker zu einem Objekt beinhaltet, dann müssen Sie die Option **Objekt mit Text verschieben** aktivieren.

Damit Sie den Objektanker überhaupt sehen, muss übrigens folgende Option eingeschaltet sein: ***Extras - Optionen - Ansicht - Seitenlayoutoptionen - Objektanker***.

Es gibt allerdings eine interessante Methode, welche es dem Benutzer dennoch sehr schwer macht, eine Shape zu verschieben. Dazu schauen Sie sich bitte die nächsten zwei Kapitel an.

Manchmal kommt auch die Frage auf, ob es möglich ist, ein Objekt so einzubringen, dass der Benutzer es nicht mehr löschen kann. Das ist in MS Word so nicht möglich. Wenn Sie trotzdem auf eine Lösung in dieser Richtung fixiert sein sollten, dann empfehle ich Ihnen nochmals das Studium der beiden folgenden Kapitel.

Eine weitere Möglichkeit bestünde darin, das Dokument in verschiedene Abschnitte zu unterteilen (*Einfügen - Manueller Wechsel - Abschnittwechsel - Fortlaufend*) und dann selektiv einzelne Abschnitte zu schützen (*Extras - Dokumentschutz - Formulare - Abschnitte*). Damit können Sie das Dokument aber ausschliesslich in Streifen unterteilen, wobei einige dieser Streifen vor jeglichem Zugriff geschützt wären.. Bedenken Sie dabei aber, dass MS Word, wenn auch nur ein einzelner Abschnitt in einem Dokument gesperrt ist, in seiner Funktionalität stark eingeschränkt wird.

Objekte, welche sich auf jeder Seite wiederholen in das Dokument einbringen

▲ Zurück zum Anfang

Geht es Ihnen darum, eine Grafik oder ein anderes Objekt in ein Dokument einzubringen, welche auf jeder Seite an der selben Stelle angezeigt werden soll, dann können Sie in MS Word mit der Kopfzeile arbeiten. Dabei sollten Sie wissen, dass Sie die Grafik mit dieser Technik auch ausserhalb des Kopfzeilenbereiches platzieren können. Wichtig dabei ist allerdings, dass die **Verankerung** der Grafik in eben dieser Kopfzeile geschieht. Der Clou ist also, dass der Anker in der Kopfzeile ist - damit ist sicher gestellt, dass sich das Objekt auf jeder Seite wiederholt - die Grafik selbst aber an einer beliebigen Stelle im druckbaren Bereich der Dokumentenseite sein kann.

Wenn Sie sich nun noch in Erinnerung rufen, das Word es erlaubt, für die erste Seite des Dokumentes eine andere Kopfzeile zu gestalten bzw. die Kopfzeile für gerade- und ungerade Seiten individuell einzurichten, dann haben Sie ein flexibles Werkzeug zur Gestaltung des Layouts in der Hand.

Wenn Sie in Ihrem Dokument mit verschiedenen Abschnitten arbeiten und die Kopfzeile in diesen Abschnitten von einander entkoppeln, nimmt die Flexibilität sogar noch zu.

Um die diskutierte Methode anzuwenden, gehen Sie wie folgt vor:

1. Wählen Sie vom Hauptmenü in Word den Befehl ***Ansicht - Kopf- und Fusszeile***.
2. Wählen Sie nun den Befehl Einfügen - Grafik - Aus Datei....
3. Der nachfolgende Schritt ist davon abhängig, welche Word-Version bei Ihnen eingesetzt wird:
   - **Word 97:** Achten Sie darauf, dass die Option ***Über den Text legen*** aktiviert ist. Wählen Sie nun die gewünschte Grafik und fügen Sie die Grafik in der Kopfzeile ein.
   - **Word2000:** Wählen Sie die gewünschte Grafik und fügen Sie die Grafik in der Kopfzeile ein. Führen Sie anschliessend einen **Rechtklick** auf die Grafik aus. Wählen Sie im **Kontextmenü** den Befehl ***Grafik formatieren...***, wechseln Sie dann in den Reiter **Layout** und aktivieren Sie dort eine Umbruchart **anders als Mit Text in Zeile**.
4. Immer noch in der Kopfzeilenansicht verschieben Sie die Grafik aus dem Kopfzeilenbereich und platzieren das Objekt im Dokument an die gewünschte Stelle. Sie werden erkennen, dass der Anker der Grafik dabei in der Kopfzeile verharrt.
5. Schliessen Sie nun die Kopfzeilenansicht. Erstellen Sie zwei oder drei Seiten und schauen Sie sich das Resultat in der Seitenlayout-Ansicht an.

Für Leute, welche mit MS Word noch nicht soviel Erfahrung haben folgt hier nun eine Anleitung, welche die Verfahren aufzeigt, um die Kopf- oder die Fusszeile für verschiedene Teile des Dokumentes individuell zu gestalten.

Erste Seite anders

Möchten Sie für die **erste Seite** Ihres Dokumentes eine Kopf- oder Fusszeile, welche vom übrigen Dokument abweicht, gehen Sie bitte wie folgt vor:

1. Rufen Sie im Word-Hauptmenü den Dialog ***Datei - Seite einrichten...*** auf.
2. Wechseln Sie in den Reiter **Seitenlayout**.
3. Aktivieren Sie im Bereich Kopf- und Fusszeile die Option **Erste Seite anders**.

Wenn Sie dieses Vorgehen befolgt haben und nun in die Kopf-/Fusszeilen-Ansicht wechseln, werden Sie entzückt feststellen, dass Ihnen nun zwei solche Bereiche zur Verfügung stehen. Auf der ersten Seite ist der Bereich mit **Erste Kopfzeile** deklariert, auf den Folgeseiten mit **Kopfzeile**. Das selbe trifft selbstverständlich auch auf die Fusszeile zu.

Gerade-/ungerade Seiten anders

Möchten Sie unterschiedliche Kopf- oder Fusszeile für die geraden bzw. die ungeraden Seiten Ihres Dokumentes, dann gehen Sie bitte wie folgt vor:

1. Rufen Sie im Word-Hauptmenü den Dialog ***Datei - Seite einrichten...*** auf.
2. Wechseln Sie in den Reiter **Seitenlayout**.
3. Aktivieren Sie im Bereich Kopf- und Fusszeile die Option **Gerade-/ungerade Seiten anders**.

Wenn Sie dieses Vorgehen befolgt haben und nun in die Kopf-/Fusszeilen-Ansicht wechseln, werden Sie entzückt feststellen, dass Ihnen nun zwei solche Bereiche zur Verfügung stehen. Alternierend ist der Bereich mit **Ungerade Kopfzeile** bzw. **Gerade Kopfzeile** deklariert. Das selbe trifft selbstverständlich auch auf die Fusszeile zu. Die Optionen *Erste Seite anders* und *Gerade-/Ungerade Seiten* lassen sich kombinieren.

Das Dokument in Abschnitte unterteilen und die Kopf-/Fusszeilen dieser Abschnitte entkoppeln

Volle Flexibilität erhalten Sie dann, wenn Sie den Aufwand auf sich nehmen, das Dokument gezielt in diverse Abschnitte zu unterteilen. Eine solche Unterteilung soll im Bezug auf die Struktur des Dokumentes Sinn machen und könnte etwa wie folgt aussehen:

- Die erste Seite stellt ein Unikum dar.
- Die Seiten 2 - 7 enthalten das Inhaltsverzeichnis des Dokumentes.
- Die Seiten 8 - 100 enthalten die eigentliche Substanz des Dokumentes.
- Die letzten Seiten des Dokumentes enthalten das Stichwortverzeichnis.

Um dieses oder eine ähnliche komplexe Struktur umzusetzen, sollten Sie das Dokument zuerst in Abschnitte unterteilen. Gehen Sie dabei wie folgt vor:

1. Stellen Sie die Einfügemarke an den Ort, an dem ein neuer Abschnitt beginnen soll.
2. Entfernen Sie den manuellen Seitenumbruch, falls ein solcher vorhanden ist.
3. Vom Word-Hauptmenü wählen Sie nun den Befehl Einfügen - Manueller Wechsel... - Abschnittswechsel - Nächste Seite.
4. Wiederholen Sie diesen Vorgang für die anderen Abschnitte.

Potentiell haben Sie nun für jeden Abschnitt einen eigenen Kopf- und Fusszeilenbereich zur Verfügung. Wahlweise können Sie bestimmen dass die Kopf- oder Fusszeile vom vorhergehenden Abschnitt übernommen wird, oder aber für einen bestimmten Abschnitt eine individuelle Kopf- oder Fusszeile gestalten. Diese Möglichkeit des Entkoppeln, funktioniert übrigens für die Kopf- oder Fusszeile unabhängig.

Standardmässig sind die Kopf-/Fusszeilen nicht entkoppelt. Um dies zu erreichen, gehen Sie bitte wie folgt vor:

1. Beginnen Sie am besten ganz am Anfang des Dokumentes.
2. Wählen Sie vom Word-Hauptmenü den Befehl ***Ansicht - Kopf- und Fusszeile...***.
3. Blättern Sie nun in der Kopf-/Fusszeilenansicht auf jene Seite, welche den nächsten neuen Abschnitt Ihres Dokumentes darstellt. MS Word zeigt Ihnen an, in welchem Abschnitt, Sie sich gerade befinden. Typischerweise steht nun dort Kopfzeile **-Abschnitt 1- ..... Wie vorherige**.
4. Schalten Sie nun den Schaltknopf mit der Bezeichnung **Wie vorherige** auf der kontextbezogenen Symbolleiste aus. Die Indikation Wie vorherige verschwindet damit.
5. Gestalten Sie nun die Kopfzeile des zweiten Abschnittes nach Ihren Wünschen. Die Kopfzeile des ersten Abschnittes ist nun nicht mehr von dieser Änderung tangiert. Die im zweiten Abschnitt gestaltete Kopfzeile gilt ab diesem Abschnitt.
6. Sollte auch die Kopfzeile des dritten Abschnittes individuell gestaltet werden, wiederholen Sie den so eben beschriebenen Vorgang in analoger Art und Weise.

Ein Wasserzeichen in das Dokument einbringen

▲ Zurück zum Anfang

Auf sehr ähnliche Art und Weise können Sie auch ein Wasserzeichen in das Dokument einbringen. Gehen Sie dabei wie folgt vor:

1. Wählen Sie vom Hauptmenü in Word den Befehl ***Ansicht - Kopf- und Fusszeile***.
2. Wählen Sie nun den Befehle ***Einfügen - Grafik - Aus Datei...***.
3. Der nachfolgende Schritt ist davon abhängig, welche Word-Version bei Ihnen eingesetzt wird:
   - **Word 97:** Achten Sie darauf, dass die Option ***Über den Text legen*** aktiviert ist. Wählen Sie nun die gewünschte Grafik und fügen Sie die Grafik in der Kopfzeile ein.
   - **Word2000:** Wählen Sie die gewünschte Grafik und fügen Sie die Grafik in der Kopfzeile ein. Führen Sie anschliessend einen **Rechtsklick** auf die Grafik aus. Wählen Sie im **Kontextmenü** den Befehl ***Grafik formatieren...***, wechseln Sie dann in den Reiter **Layout** und aktivieren Sie dort eine Umbruchart **anders** *als Mit Text in Zeile*.
4. Wählen Sie nun die gewünschte Grafik und fügen Sie die Grafik in der Kopfzeile ein.
5. Immer noch in der Kopfzeilenansicht verschieben Sie die Grafik aus dem Kopfzeilenbereich und platzieren das Objekt im Dokument an die gewünschte Stelle. Sie werden erkennen, dass der **Anker** der Grafik dabei in der Kopfzeile verharrt.
6. Führen Sie einen **Rechtsklick** auf die Grafik aus und wählen Sie im Kontextmenü den Befehl ***Grafik formatieren...***.
7. Wechseln Sie in das Reiter **Grafik**.
8. Erhöhen Sie nun **Helligkeit** und **Kontrast** der Grafik.
9. Schliessen Sie nun die Kopfzeilenansicht und schauen Sie sich das Resultat in der Seitenlayout-Ansicht an.

Dieses Wasserzeichen wird sich übrigens auf jeder Seite wiederholen.

VBA-technischer Umgang mit «Shapes» und «InlineShapes»

▲ Zurück zum Anfang

Einleitung

Will man eine Shape oder eine InlineShape mit VBA ansprechen, dann sind vor allem die folgenden Punkte zu beachten:

- Die Tatsache, dass Word zwischen Shapes und InlineShapes unterscheidet, muss sich auch im VBA-Code widerspiegeln. Wählen Sie also das richtige Objekt. Beachten Sie auch, dass sich die Syntax unterscheiden kann und denken Sie auch daran, dass Eigenschaften und Methoden von Shapes und InlineShapes nicht in allen Punkten deckungsgleich sind.
- Wie die meisten Objekte können Sie Shapes und InlineShapes auf zwei Arten ansprechen, nämlich über einen sequentiellen Index, wobei das erste Objekt im Dokument den Index «1» trägt, das zweite Objekt den Index «2» u.s.w.
- Sie können eine Shape (und nur eine Shape) aber auch direkt ansprechen, nämlich über den Namen des Objektes. Nun ist es aber so, dass dieser Name intern von Word vergeben wird, wenn das Objekt in das Dokument eingebracht wird. In der Regel kennen Sie diesen Namen also nicht, Sie können Ihn bestenfalls erahnen.
- Wesentlich bessere Karten haben Sie, wenn Sie die Shape oder die InlineShape ebenfalls mit VBA erstellen, dann können Sie nämlich einen eindeutigen Namen nach Ihrem Geschmack vergeben und das Objekt später über diesen Namen jederzeit ansprechen.
- Weiter gilt es zu berücksichtigen, dass Word intern ein Dokument nicht als Stück aus einem Guss betrachtet. Vielmehr kennt Word genau elf StoryTypes, also z.B. der Hauptteil des Dokumentes, Kopf-/Fusszeilenbereich, Textfelder etc. Sie finden am Ende dieses Skriptes eine Tabelle, welche sämtliche StoryTypes, welche MS Word definiert auflistet und wie diese Story-Ranges VBA-technisch anzusprechen sind.

Shapes

Folgendes Beispiel spricht die erste Shape im Hauptteil des Dokumentes an und vergrössert Länge und Breite dieses Objektes um 25%:

```
Sub ShapeImHauptteil()
  Dim Form As Shape
  Set Form = ActiveDocument.Shapes(1)
  Form.Height = Form.Height * 1.25
  Form.Width = Form.Width * 1.25
End Sub
```

(Eine Shape wird angesprochen und formatiert.)

Folgendes Beispiel spricht die erste Shape, welche sich im Kopfzeilenbereich des ersten Abschnittes befindet an. Bedenken Sie dabei, dass es neben der primären Kopfzeile allenfalls noch eine *«FirstPage-Kopfzeile»* und eine *«EvenPage-Kopfzeile»* geben kann.

```
Sub ShapeInDerKopfzeile()
  Dim Form As Shape
  Set Form = _
     ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Shapes(1)
  Form.Height = Form.Height * 1.25
  Form.Width = Form.Width * 1.25
End Sub
```

(Eine Shape, welche sich in der Kopfzeile befindet, wird angesprochen und formatiert.)

Dieses Beispiel führt nun verständlicherweise zur Annahme, dass man die Konstante mit dem Namen `wdHeaderFooterPrimary` bei Bedarf einfach mit den Konstanten `wdHeaderFooterFirstPage` oder `wdHeaderFooterEvenPages` austauschen könne, um zu erreichen, dass die Shape in die Kopfzeile der ersten Seite bzw. der geraden Seiten eingefügt wird. Die ist aber aus Gründen, welche mir auch nicht ganz klar sind, nicht möglich. Der Code läuft zwar problemlos ab, stellt aber die Shape trotzig trotzdem in die primäre Kopfzeile.

Ausweg aus diesem Dilemma schafft eine zusätzliche Eigenschaft des *Shape*-Objektes nämlich die Möglichkeit der **Verankerung**. Dabei muss man wissen, dass sobald man eine Shape verankert die Positionierung nicht mehr in Relation zu den Seitenränder passiert, sondern in **Relation zum Absatz**, wo der Anker geworfen wird. Diese Tatsache muss also berücksichtigt werden.

Damit das Dokument überhaupt über die Bereiche *Kopfzeile erste Seite* bzw. *Kopfzeile ungerade Seite* verfügt, müssen diese Eigenschaften zuerst eingeschaltet werden. Dabei handelt es sich um eine Eigenschaft eines Abschnittes (*Section*). Das Beispiel geht davon aus, dass das vorliegende Dokument aus nur einem Abschnitt besteht. Diese Einstellung kann manuell oder auch programmtechnisch geschehen. Das Gleiche gilt selbstverständlich auch für die Fusszeilenbereiche.

Im folgenden Beispiel wird in der Kopfzeile der ersten Seite des Dokumentes (genau gesagt im ersten Abschnitt dieses Dokumentes) eine Linie verankert, welche als Faltmarke für den gedruckten Brief dient. Da die Anforderung sagt, dass die Marke ausschliesslich auf der ersten Seite gedruckt werden soll, kommt die besprochene Problematik zum Zuge. Das Code-Beispiel exerziert folgende Schritte durch:

- Er schaltet die Eigenschaft **Erste Seite anders** ein.
- Er berechnet die **Distanz des ersten Absatzes** im Kopfzeilenbereich der ersten Seite zum Papierrand
- Er fügt in den Kopfzeilenbereich der erste Seite eine Shape ein und **verankert** diese auch dort. Damit die Distanz zum linken Blattrand stimmt, wird der Abstand der Absatzmarke zum Papierrand abgezogen.

```
Sub FalzmarkeEinfügenNurErsteSeite()
  'Die Länge des Striches beträgt 0,9 - 0,5 = 0,4 cm
  Dim oKz As HeaderFooter, FM As Shape
  ActiveDocument.Sections(1).PageSetup.DifferentFirstPageHeaderFooter = True
  Set oKz = ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage)
  Rand = _
   oKz.Range.Paragraphs(1).Range.Information(wdHorizontalPositionRelativeToPage)
  Set FM = oKz.Shapes.AddLine(CentimetersToPoints(0.5) - Rand, _
                              CentimetersToPoints(10.4), _
                              CentimetersToPoints(0.9) - Rand, _
                              CentimetersToPoints(10.4), oKz.Range)

  FM.Line.Weight = 0.25
  FM.LockAnchor = True
End Sub
```

(Fügt eine Faltmarke in das Dokument ein. Dieser Hilfsstriche erscheint ausschliesslich auf *der ersten Seite* des Dokumentes.)

Wie schon mehrfach erwähnt, kann eine Shape (konkret ein Textfeld oder eine AutoForm - Legende) Container für weiteren Text oder Objekte sein. Auch dies muss sich im Code niederschlagen.

Folgendes Beispiel formatiert den Text im ersten Textfeldes, welches sich im Hauptteil des Dokumentes befindet, rot. Da wir nicht davon ausgehen können, dass das erste Textfeld auch die erste Shape im Dokument ist (es könnte ja z.B. noch eine Grafik vorhanden sein, welche ebenfalls eine Shape darstellen kann) berücksichtigen wir dies im Code.

```
Sub InhaltEinesTextfeldesFormatieren()
  Dim Form As Shape
  For Each Form In ActiveDocument.Shapes
    If Form.Type = msoTextBox Then
      Form.TextFrame.TextRange.Font.ColorIndex = wdRed
    End If
  Next
End Sub
```

(Der Text in einem Textfeld wird formatiert.)

Im folgenden wird eine Shape (konkret ein Textfeld) an der Stelle eingefügt, an der sich die Einfügemarke befindet. Da wir diese Shape mit VBA einbringen können wir dem Objekt einen für uns eindeutigen und bekannten Namen vergeben, um später gezielt darauf zurückzugreifen. Das Beispiel demonstriert dann auch im Anschluss, wie man eine Shape später beim Namen ansprechen kann.

```
Sub TextfeldEinbringen()
  Dim Form As Shape
  Set Form = _
     ActiveDocument.Shapes.AddTextbox(Orientation:=msoTextOrientationHorizontal, _
        Left:=CentimetersToPoints(2.5), _
           Top:=CentimetersToPoints(2.5), _
              Width:=CentimetersToPoints(6), _
                 Height:=CentimetersToPoints(4))
  Form.Name =«MeinErstesTextfeld"
End Sub

Sub TextfeldBeimNamenAnsprechen()
  ActiveDocument.Shapes("MeinErstesTextfeld").TextFrame.TextRange.Text = _
    «Hallo Welt"
End Sub
```

(Ex VBA können Sie einer Shape einen Namen Ihrer Wünsche verpassen um später die Form bei Namen anzusprechen. Achten Sie auf die Eindeutigkeit dieser Namen.

Folgender Code wandelt die erste Shape im Hauptteil des Dokumentes in eine InlineShape um: Beachten Sie, dass nicht jede Art von Shapes in eine InlineShape umgewandelt werden kann. Die Tabelle, weiter oben im Skript, gibt darüber Auskunft.

```
Sub ShapeNachInlineShape()
  ActiveDocument.Shapes(1).ConvertToInlineShape
End Sub
```

(Eine Shape wird zu einer InlineShape.)

Zum Schluss noch ein Beispiel aus der Praxis: Ein Kunde stand vor dem Problem, dass Hunderte von Dokumenten, welche die ein und die gelbe Grafik enthielten, geändert werden mussten. Diese Grafik sollte auf die selbe Art und Weise in jedem der Dokumente programmtechnisch geändert werden. Man konnte nicht davon ausgehen, dass es sich dabei immer um die erste Shape im Dokument handelte. Der interne Namen, welcher Word für diese Shape vergab, entpuppte sich - nicht unerwartet - als zufällig.

In solch einem Fall hilft wohl nur eine Art Rasterfahndung. Überlegen Sie sich, was denn bei den Objekten wirklich eindeutig ist und vergleichen Sie jede der Shapes im Dokument (bzw. so lange bis sie identifiziert ist) mit diesem Raster.

Im folgenden Code wird im Hauptteil des Dokumentes nach einer Shape (konkret eine Grafik) gefahndet, welche genau 320 x 240 Pixel misst. Wird diese Grafik identifiziert, wird die Helligkeit der Grafik intensiviert.

```
Sub RasterFahndung()
  Dim Form As Shape
  Flag = False
  For Each Form In ActiveDocument.Shapes
    If Form.Type = msoPicture Or Form.Type = msoLinkedPicture Then
      If Form.Width = 320 And Form.Height = 240 Then
        Flag = True
        Form.PictureFormat.Brightness = 0.7
        Exit For
      End If
    End If
  Next
  If Flag = False Then
    MsgBox«Keine Entsprechung gefunden. - Bitte Interpol einschalten!"
  End If
End Sub
```

(Enthält ein Dokument viele Shapes, besteht die Herausforderung darin, die richtige zu finden. Oft ist dann Rasterfahndung angesagt)

Selbstverständlich können Sie auch andere bzw. zusätzliche Kriterien einbeziehen.

Die Sache gelingt wohl nur dann gut, wenn Sie Ihre zu bearbeitenden Dokumente sehr systematisch und detailgetreu aufgebaut haben. Schon kleine (auch nicht sichtbare) Unterschiede können das Unterfangen zum Scheitern bringen.

InlineShape

Folgendes Beispiel spricht die erste InlineShape im Hauptteil des Dokumentes an und vergrössert Länge und Breite dieses Objektes um 25%:

```
Sub InlineShapeImHauptteil()
  Dim Form As InlineShape
  Set Form = ActiveDocument.InlineShapes(1)
  Form.Height = Form.Height * 1.25
  Form.Width = Form.Width * 1.25
End Sub
```

(Eine InlineShape, welche sich in der Hauptteil des Dokumentes befindet, wird angesprochen und formatiert.)

Folgendes Beispiel spricht die erste InlineShape, welche sich im Kopfzeilenbereich des ersten Abschnittes befindet an. Bedenken Sie dabei, dass es neben der primären Kopfzeile allenfalls noch eine *«FirstPage-Kopfzeile»* und eine *«EvenPage-Kopfzeile»* geben kann.

```
Sub InlineShapeInDerKopfzeile()
  Dim oDoc As Document, Form As InlineShape
  Set oDoc = ActiveDocument
  Set Form = _
     oDoc.Sections(1).Headers(wdHeaderFooterPrimary).Range.InlineShapes(1)
  Form.Height = Form.Height * 1.25
  Form.Width = Form.Width * 1.25
End Sub
```

(Eine InlineShape, welche sich in der Kopfzeile befindet, wird angesprochen und formatiert.)

Im folgenden wird eine Grafik als InlineShape an der Stelle eingefügt, an der sich die Einfügemarke befindet.

```
Sub InlineShapeEinbringen()
  Dim Form As InlineShape
  Set Form = ActiveDocument.InlineShapes.AddPicture( _
     FileName:="C:\Windows\Kreise.bmp", _
        LinkToFile:=False, _
           Range:=Selection.Range)
  Form.PictureFormat.Contrast = 0.2
End Sub
```

(Eine InlineShape, wir in das Dokument eingebracht.)

Bitte beachten Sie, dass Sie einer InlineShape im Gegensatz zu einer Shape keinen Namen vergeben können. Auch Word vergibt dieser Art von Objekten keinen Namen. Eine InlineShape liegt auf der selben Ebene, wie der Text; sie ist integrierter Bestandteil des Textes. Sie können die InlineShape aber jederzeit über den Index ansprechen:

```
Sub InlineShapeAnsprechen()
  Dim Form As InlineShape
  ActiveDocument.InlineShapes(1).PictureFormat.Brightness = 0.3
End Sub
```

(InlineShapes über ihren Index ansprechen.)

Neben der Identifikation über den Index, bleiben aber gute Alternativen. So können Sie einer InlineShape eine Textmarke mit einem eindeutigen Namen vergeben.

Dazu **markieren** Sie die InlineShape und wählen vom Hauptmenü den Dialog ***Einfügen - Textmarke*** und vergeben einen Namen z.B. Firmenlogo. Nun können Sie diese InlineShape wie folgt ansprechen:

```
Sub InlineShapeAnsprechenTextmarke()
  Dim Form As InlineShape
  If ActiveDocument.Bookmarks.Exists("Firmenlogo") Then
    Set Form = ActiveDocument.Bookmarks("Firmenlogo").Range.InlineShapes(1)
  End If
End Sub
```

(Weisen Sie einer InlineShapes eine Textmarke zu, um sie programmtechnisch zu finden.)

Allenfalls lässt sich die gesuchte InlineShape auch auf Grund ihres Typs identifizieren und das geht dann z.B. so:

```
Sub InlineShapeNachTypIdentifizieren()
  Dim ILS As InlineShape
  For Each ILS In ActiveDocument.InlineShapes
    Select Case ILS.Type
     Case wdInlineShapePicture, wdInlineShapeLinkedPicture
       ILS.Select
       Exit For
    End Select
  Next
End Sub
```

(Findet das erste *Bild* im Dokument, welches als InlineShape eingebracht wurde.)

Gewisse InlineShapes verfügen über die Eigenschaft **OLEFormat**. Diese Objekte verfügen über ein erweitertes Objekt-Modell wodurch es möglich ist, dem Objekt einen eindeutigen Namen zu zu weisen.

Diese Option ist aber nicht bei allen Typen von InlineShapes gegeben. Im folgenden finden Sie den Code, welcher eine *ActiveX-Control* (eingefügt über die Symbolleiste *Steuerelement-Toolbox*) auf Grund des eindeutigen Namens finden kann.

```
Sub ActiveXControlBeimNamenIdentifizieren()
  Dim ILS As InlineShape
  For Each ILS In ActiveDocument.InlineShapes
    If ILS.Type = wdInlineShapeOLEControlObject Then
      If ILS.OLEFormat.Object.Name = "CheckBox1" Then
        ILS.Select
        Exit For
      End If
    End If
  Next
End Sub
```

(Findet die *ActiveX-Control* mit dem Namen *CheckBox1*, welche als InlineShape über die Symbolleiste *Steuerelement-Toolbox* in das Dokument eingebracht wurde..)

Folgender Code wandelt die erste InlineShape im Hauptteil des Dokumentes in eine Shape um:

```
Sub InlineShapeNachShape()
  ActiveDocument.InlineShapes(1).ConvertToShape
End Sub
```

(Aus einer InlineShape wird eine Shape.)

Eine Shape (konkret ein Textfeld) kann eine InlineShape beinhalten. Das folgende Beispiel geht davon aus, dass die erste Shape im Hauptteil des Dokumentes ein Textfeld ist und dass sich in diesem Textfeld eine InlineShape befindet, nämlich eine Grafik. Diese Grafik wird im folgenden Code verkleinert:

```
Sub InlineShapeInTextfeld()
  Dim Grafik As InlineShape
  Set Grafik = _
     ActiveDocument.Shapes(1).TextFrame.TextRange.InlineShapes(1)
  Grafik.Height = Grafik.Height * 0.66
  Grafik.Width = Grafik.Width * 0.66
End Sub
```

(Eine InlineShape, welche sich im ersten Textfeldes des Dokumentes befindet, wird angesprochen und formatiert.)

Textfelder und Tabellenzellen können übrigens ausschliesslich InlineShapes beherbergen. *«Shape in Shape»* scheint also keine mögliche Konstellation darzustellen.

Tabelle der *«StoryRanges»* (Dokumententeile) in MS Word

▲ Zurück zum Anfang

Komponente	Word-Objekt	Relation	Bemerkung	Beispiel
Haupttext	MainTextStory	1 pro Dokument	MainTextStory und Dokument können synonym behandelt werden.	`ActiveDocument.Fields.Update`
Hauptkopfzeile	PrimaryHeaderStory	1 pro Abschnitt (Section)	Ist immer vorhanden, kann aber auch leer sein. Kann auch der OddPage-HeaderStory entsprechen, welche es nominell nicht gibt	`ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Fields.Update`
Kopfzeile für erste Seite	FirstPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Erste Seite anders" aktiviert ist	`ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage).Range.Fields.Update`
Kopfzeile für gerade Seite	EvenPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Gerade-/ungerade Seiten anders" aktiviert ist.	`ActiveDocument.Sections(1).Headers(wdHeaderFooterEvenPages).Range.Fields.Update`
Hauptfusszeile	PrimaryFooterStory	1 pro Abschnitt (Section)	siehe Hauptkopfzeile	`ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range.Fields.Update`
Fusszeile für erste Seite	FirstPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterFirstPage).Range.Fields.Update`
Fusszeile für gerade Seite	EvenPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben	`ActiveDocument.Sections(1).Footers(wdHeaderFooterEvenPages).Range.Fields.Update`
Kommentare	CommentsStory	0 zu n pro Dokument	Kommentare können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Comments(1).Range.Fields.Update`
Endnote	EndnotesStory	0 oder 1 pro Dokument	Endnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Endnotes(1).Range.Fields.Update`
Fussnoten	FootnotesStory	0 zu n pro Dokument	Fussnoten können nur Objekte (Elemente) der MainTextStory sein	`ActiveDocument.Footnotes(1).Range.Fields.Update`
Textfeld	TextFrameStory	0 zu n pro Story	Textfelder sind in erster Linie Objekte (Elemente) von Shapes	`ActiveDocument.Shapes(1).TextFrame.TextRange.Fields.Update`

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Word findet bei der Generierung des Abbildungsverzeichnisses keine Einträge

Cindy Meister

► Positioning Graphics - Placing Pictures Where You Want Them in Word

Microsoft

Deutsch

► WORD 97: Grafiken standardmäßig in Zeichnungsebene eingefügt
► WORD2000: Bild kann nicht als "Über den Text legen" eingefügt werden

Englisch

► WD97: General Information About Floating Objects
► WD2000: General Information About Floating Objects

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Shapes.htm

René Probst, Oktober 2000

Änderungen

- 21. November 2000 - InlineShapes nach Typus identifizieren
- 21. November 2000 - OLEFormat und Name-Eigenschaft
- 21. November 2000 - Shape in FirstPageHeader einfügen

Wie kann ich auf die Adressen im Outlook Kontakte-Ordner zugreifen?

▲ Zurück zum Hauptindex

Ausgangslage

> **Eine sehr praktische Word-Funktion erlaubt es, in komfortabler Manier auf den Kontakte-Ordner von Outlook 97, 98 oder 2000 zuzugreifen. Diese Funktion ist allerdings gut versteckt und muss zuerst ausgegraben werden.**
>
> **Bitte verwechseln Sie dabei Outlook aus dem Microsoft Office Paket nicht mit Outlook Express, welches mit dem Internet Explorer daherkommt. Auf das Adressbuch von Outlook Express kann nicht direkt zugreifen werden. Ungeschickt und nicht ganz ehrlich von Microsoft ist, dass *Internet Mail* nach *Outlook Express* umbenannt wurde, denn MS Outlook und Outlook Express sind technisch und lizenztechnisch gesehen, völlig verschiedene Produkte, wenn sie auch teilweise den selben Funktionsumfang aufweisen.**

Adressbuchfunktion installieren

Zuerst sollten Sie sicher stellen, dass die Adressbuchfunktion auch wirklich auf Ihrem Rechner installiert ist. Bei einer Standardinstallation von Office 97 oder Office2000 ist das meines Wissens nicht der Fall. Überprüfen Sie also unter ***Start - Einstellungen -Systemsteuerung - Software - Microsoft Office 97 - Word***, ob das **AdreßBuch** installiert ist. Falls nötig holen Sie dies nun nach.

Adressbuchfunktion einrichten

Falls Sie in Word den Zugriff zum Outlook Kontakte-Ordner einrichten möchten, gehen Sie wie folgt vor:

1. Vom Word-Hauptmenü wählen Sie den Dialog ***Extras - Anpassen.***
2. Wechseln Sie in diesem Dialog zum Reiter ***Befehle.***
3. Markieren Sie in der linken Liste **Kategorien** den Eintrag **Einfügen.**
4. Suchen Sie in der rechten Liste **Befehle** den Eintrag **Adressbuch...** (ziemlich weit unten in der Liste).
5. Mittels Drag 'n Drop-Technik ziehen Sie diesen Befehl auf eine Symbolleiste Ihrer Wahl. Damit sich das Ikon ablegen lässt, sollte es bündig mit einem bereits vorhandenen Befehl sein.
6. Schliessen Sie den Dialog mit ***Schliessen*** ab.

Adressbuchfunktion benutzen

1. Platzieren sie die Einfügemarke an die Stelle an welcher Sie eine Adresse einfügen möchten.
2. Klicken Sie nun auf das Symbol ***Adresse einfügen***.
3. Wählen Sie das Adressbuch, welches Sie benutzen möchten.
4. Wählen Sie aus der Namensliste eine Adresse aus.

Anmerkungen

- Sie finden den Befehle *Einfügen Adresse* übrigens auch im Word-Dialog ***Extras - Umschläge und Etiketten***.
- Als Alternative zum Outlook Kontakte-Ordner können Sie auch auf das **Personal Address Book** von **MS Exchange** oder auf **MS Schedule+** zugreifen. Dies bedingt natürlich, dass diese Programme ordnungsgemäss auf Ihrem Rechner installiert sind. Erfahrungsgemäss arbeitet MS Exchange zu mindest unter **Windows 98** nur sehr unzuverlässig. Meisten fangen die Probleme bereits mit der Installation dieses Programms an.
- Einer der hier eingeblendeten Links führt Sie zu einem Artikel, welcher beschreibt, wie Sie die Felder welche in der Adresse vorkommen sollen, auswählen bzw. anders anordnen können.
- Sollte sich Word beim Aufruf des Befehles *Adresse einfügen* aufhängen, könnte dies daran liegen, dass Sie das Word-Adressbuch noch gar nicht installiert haben. Wie die Adressbuchfunktion zu installieren ist, wird am Anfang dieses Kapitels beschrieben.

Links zu diesem Thema

Microsoft

► **WD97: How to Add the Insert Address Button to a Toolbar**

Eigene Links

► **Wie kann ich die Adressfelder und deren Reihenfolge, welche ich über den Befehl Einfügen Adresse in mein Dokument einfüge beeinflussen?**

► **Wieso kann ich nicht auf das Adressbuch von Outlook Express 4 oder 5 zugreifen**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/ABuch.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Das Outlook-Feld «Kategorien» steht in der Word-Seriendruckfunktion nicht zur Verfügung.

◄ Zurück zum Hauptindex

Problemstellung

> **Wenn ich einen Serienbrief erstellen möchte und das Hauptdokument mit dem Outlook-Kontakte-Ordner verbinde, suche ich vergeben nach dem für mich sehr wichtigen Outlook-Feld mit dem Namen *Kategorien*. Da ich nun aber meine Kontakte sorgfältig nach diesem Kriterium unterteilt habe, gelingt es mir nicht, einen Serienbrief mit selektiven Adressaten zu erstellen. Auch kann ich den Inhalt des genannten Feldes nicht abfragen, um klug darauf reagieren.**

Antwort

Das genannte Feld steht in Word leider tatsächlich nicht zur Verfügung, wie auch eine beachtliche Anzahl anderer Felder aus dem Outlook-Datenmodell. Um trotzdem mit diesem Feld arbeiten zu können, müssen Sie einen Zwischenschritt einschlagen und Ihre Kontakte in Outlook, wo dieses Feld existiert, vorselektionieren. Gehen Sie konkret wie folgt vor:

Lösungsvarianten

Drei Lösungsvarianten werden hier besprochen. Dabei fällt auf, dass bei allen Varianten der Lösungsansatz in MS Outlook und nicht in Word gefunden wird.

1. Sie könnten in MS Outlook die Ansicht so definieren, dass nur die Kontakte einer bestimmten Kategorie angezeigt werden und anschliessend diese Kontakte in einem eigens erstellten Kontakte-Ordner abspeichern.
2. Sie könnten in MS Outlook die Daten, welche im Feld mit dem Namen Kategorien versorgt sind in ein anderes Feld duplizieren. Stellen Sie dabei sicher, dass dieses Feld: 1. wirklich unter Word zur Verfügung steht und 2. dass dieses Feld, welches Sie bei diesem Vorgehen missbrauchen, wirklich nicht für anderes gebraucht wird.
3. MS Outlook2000 (und nur Outlook2000), bietet die Möglichkeit einen Serienbrief aus Outlook zu initialisieren anstatt aus Word.

Datensätze einer bestimmten Kategorie in einen eigens erstellten Kontakte-Ordner duplizieren

Lösungsweg

Zusammenfassend sollten Sie bei diesem Ansatz folgende Schritte nachvollziehen:

1. Benutzen Sie in MS Outlook einen **Filter**, um nur die Adressen einer bestimmten Kategorie anzuzeigen.
2. Erstellen Sie einen eigenen benutzerdefinierten Kontakte-Ordner.
3. Kopieren Sie die filtrierten Einträge in diesen Ordner.
4. Verfahren Sie mit den anderen Kategorien ebenso.
5. Zurück in Word werden Ihnen jetzt mehrere Kontakte-Ordner angezeigt. Sie können diese nun also selektiv verarbeiten.

Vorgehensweise

Leider ist die Vorgehensweise um einen Filter zu definieren, in jeder MS Outlook Version etwas anders. Ich beschränke mich deshalb darauf, das Verfahren für **Outlook2000** Schritt für Schritt zu erklären. Verwenden Sie entweder einer der unten eingeblendeten Links oder die Online-Hilfe von Outlook um zu erfahren, wie in der Version mit der Sie arbeiten einen Filter eingerichtet wird und das Ergebnis in einen neuen Ordner kopiert werden kann. Beginnen Sie bei der Suche mit dem Punkt **Ansicht** im Hauptmenü von Outlook.

Die folgende Beschreibung geht davon aus, dass Sie unter anderem eine Kategorie **Privat** eingerichtet haben und die Einträge dieser Kategorie herausfiltern möchten.

1. Starten Sie **Outlook2000** und wechseln Sie zum **Kontakte-Ordner**.
2. Vom Hauptmenü in Outlook2000 wählen Sie den Dialog ***Ansicht - Aktuelle Ansicht - Ansicht anpassen***.
3. Im erscheinenden Dialog wählen Sie die Schaltfläche ***Filtern...*** und wechseln in den Reiter **Erweitert**.
4. Klicken Sie auf die Schaltfläche ***Feld*** und wählen Sie dann ***Häufig verwendete Felder - Kategorien***.
5. Stellen Sie im Kombinationsfeld **Bedingung:** den Wert **ist (genau)** ein.
6. Schreiben Sie in das Feld **Wert:** den Wert **Privat**.
7. Schliessen Sie den Dialog mit ***Ok*** (2x) ab. Es werden Ihnen nun nur noch die Kontakte der Kategorie "Privat" angezeigt.
8. Wählen Sie nun vom Hauptmenü den Dialog ***Datei - Neu - Ordner***.
9. Achten Sie darauf, dass dieser neuer Ordner vom Typus **Kontakte** ist und geben Sie dem Ordner z.B. den Namen **Privat**.
10. Wählen Sie nun vom Hauptmenü ***Bearbeiten - Alles markieren***.
11. Wichtig: Halten Sie die Taste **[Strg]** gedrückt während Sie sämtliche markierten Adressen in der Anzeige in den neuen Ordner ziehen.
12. Für die weiteren Kategorien beginnen Sie bitte wieder ganz oben und gehen analog vor.
13. Kehren Sie in die Word-Anwendung zurück und rufen Sie dort den Dialog **Extras - Seriendruck... - Datei importieren - Adressbuch verwenden...- Outlook Adressbuch verwenden..** auf. Der neue Ordner **Privat** und die weitere Ordner, welche Sie allenfalls erstellt haben, sollten nun in der Liste wählbar sein.
14. Sie können nun Kategorie um Kategorie in Ihrem Sinne verarbeiten.

Die Daten im Feld Kategorien in ein anders Feld duplizieren

Lösungsweg

Zusammenfassend sollten Sie bei diesem Ansatz folgende Schritte nachvollziehen:

1. Sie suchen und bestimmen ein Outlook Feld im Kontakte-Ordner, welches in Word zur Verfügung steht von Ihnen aber nicht (auch nicht zukünftig) gebraucht wird..
2. Entscheiden Sie, ob diese Manipulation in Ihrem Outlook-Kontakte-Ordner selbst, oder in einer Kopie dieses Ordners geschehen soll.
3. Da Sie diese Duplizierung vermutlich für alle Datensätze im Kontakte-Ordner machen müssen, drängt sich hier eine automatisierte Lösung an.

Vorgehensweise

Nachfolgendes Programm nimmt Ihnen diese Arbeit also ab. Das Makro bietet fünf alternative Felder zur Auswahl an. Alle diese fünf Felder stehen in Word zur Verfügung werden aber wohl eher selten gebraucht. Es liegt an Ihnen, welches dieser Felder Sie opfern möchten.

Im Eingangsdialog des Programms können Sie entscheiden, ob die Manipulation im Standard-Kontake-Ordner direkt geschehen soll, oder ob das Programm die Datensätze in einen Behelfsordner kopieren soll, um die Kontakte dort zu manipulieren.

Übrigens, Dank dem Einsatz von "Office-Automation" können Sie dieses Makro ex Word starten. Die Outlook-Anwendung wird dann im Hintergrund mittels VBA gestartet. Kopieren Sie also den gesamten Code, welcher unten abgebildet ist, in ein neues Modul im VBA-Editor von Word und sichern Sie allenfalls vorgängig die ***.pst-Datei**, welche unter anderem auch Ihre Kontakte beherbergt.

Wie schon angedeutet, ist das Programm dialoggeführt. Eine weiterreichende Erklärung erübrigt sich deshalb.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
' Diese Routine kopiert die Werte des Feldes "Kategorien" für alle Datensätze
' im Outlook Kontakte-Ordner in ein anderes Feld, welches auch in einem
' Word-Seriendruck verfügbar ist. Sie können zwischen fünf Zielfelder wählen.
' Diese Aktion ist notwendig, da das Datenmodell von Microsoft Outlook nicht
' sauber in Word abgebildet wird.
' Das Feld "Kategorien" steht nämlich in Word nicht zur Verfügung.
'
' Erstellt am 10/10/1999 von René Probst / eku / CH-8152-Glattbrugg

Private asVisible As Boolean
Private blTitel As String
Private blText As String
Private oBlase As Balloon
Private Feld(5, 2) As Variant
Private Ordner As String
Private Anzahl As Long
Private Wahl As Long

Sub OutlookFeldKategorienKopieren()
  Set oAssistant = Assistant
  If oAssistant Is Nothing Then
    MsgBox "Der Office-Assistent ist nicht oder nicht korrekt installiert.", _
       vbCritical, blTitel
    End
  End If
  Set oAssistant = Nothing
  asVisible = Assistant.Visible
  Set oBlase = Assistant.NewBalloon
  blTitel = "Outlook-Feld " & Chr(34) & "Kategorien" & Chr(34) & " kopieren"
  Assistant.Visible = True
  Assistant.Animation = msoAnimationGreeting
  Feld(1, 1) = "Weitere_Region"
  Feld(2, 1) = "Weiterer_Staat"
  Feld(3, 1) = "Funkruf_Piepser"
  Feld(4, 1) = "Sekretariat"
```

```
  Feld(5, 1) = "PartnerIn"
  Feld(1, 2) = "OtherAddressState"
  Feld(2, 2) = "OtherAddressCountry"
  Feld(3, 2) = "PagerNumber"
  Feld(4, 2) = "AssistantName"
  Feld(5, 2) = "Spouse"
  blText = "So wird's gemacht" & vbCrLf & vbCrLf
  blText = blText & "Wählen Sie das Feld, welche die Werte im Feld "
  blText = blText & Chr(34) & "Kategorien" & Chr(34) & " aufnehmen soll."
  blText = blText & vbCrLf & vbCrLf & "Geben Sie anschliessend an, "
  blText = blText & "ob das Resultat in einem neuen Ordner oder aber im "
  blText = blText & "Standard-Kontakte-Ordner gespeichert werden soll. "
  blText = blText & "Falls Sie den Standard-Kontakte-Ordner angeben, dann wird "
  blText = blText & "der Inhalt des von Ihnen gewählten Feldes überschrieben."
  Wahl = BlaseAnzeigen1
  Select Case Wahl
    Case -2
      HappyEnd
    Case 1 To 5
      OutlookFeldKategorienSchleife
      BlaseAnzeigen2
      HappyEnd
  End Select
End Sub

Private Sub OutlookFeldKategorienSchleife()
  Dim olAnw As Outlook.Application
  Dim olNS As Outlook.NameSpace
  Dim olOrdner As Outlook.MAPIFolder
  Set olAnw = New Outlook.Application
  Set olNS = olAnw.GetNamespace("MAPI")
  Set olOrdner = olNS.GetDefaultFolder(olFolderContacts)
  If Not oBlase.Checkboxes(1).Checked Then
    Assistant.Animation = msoAnimationWorkingAtSomething
    Dim olNeu As Outlook.MAPIFolder
    Ordner = "SB" & Format(Date, "yyyymmdd") & Format(Time, "hhmmss")
    Set olNeu = olOrdner.Folders.Add(Ordner)
    i = 0
    For Each DS In olOrdner.Items
      i = i + 1
      If i Mod 10 = 0 Then Report 1, i
      Set Buffer = DS.Copy
      Buffer.Move olNeu
    Next
    Set olOrdner = olNeu
  End If
  Assistant.Animation = msoAnimationWorkingAtSomething
  i = 0
  For Each DS In olOrdner.Items
    i = i + 1
    If i Mod 10 = 0 Then Report 2, i
    Select Case Wahl 'Na ja, ich würde es ja gerne dynamischer machen, aber wie?!
      Case 1: If DS.Categories <> "" Then DS.OtherAddressState = DS.Categories
```

```
      Case 2: If DS.Categories <> "" Then DS.OtherAddressCountry = DS.Categories
      Case 3: If DS.Categories <> "" Then DS.PagerNumber = DS.Categories
      Case 4: If DS.Categories <> "" Then DS.AssistantName = DS.Categories
      Case 5: If DS.Categories <> "" Then DS.Spouse = DS.Categories
    End Select
    DS.Save
  Next
  If i = 0 Then KeineDatensätze
  Anzahl = i
  Assistant.Animation = msoAnimationCharacterSuccessMajor
End Sub

Private Sub KeineDatensätze()
  MsgBox "Der Outlook-Kontakte-Ordner enthält keine Datensätze.", vbOKOnly, blTitel
  HappyEnd
End Sub

Private Sub Report(tArt, tAnz)
  Select Case tArt
    Case 1
      tText = tAnz & " Datensätze kopiert. - Bitte warten..."
    Case 2
      tText = tAnz & " Datensätze verarbeitet. - Bitte warten..."
  End Select
  StatusBar = tText
  DoEvents
End Sub

Private Sub HappyEnd()
  Set olAnw = Nothing
  Set olNS = Nothing
  Set olOrdner = Nothing
  Set olNeu = Nothing
  Assistant.Visible = asVisible
  StatusBar = ""
  End
End Sub

Private Function BlaseAnzeigen1()
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = blTitel
    .Text = blText
    For i = 1 To 5
      .Labels(i).Text = Feld(i, 1)
    Next i
    .Checkboxes(1).Text = "Direkt im Standard-Kontakte-Ordner speichern."
    .Button = msoButtonSetCancel
    BlaseAnzeigen1 = .Show
  End With
End Function
```

```
Private Function BlaseAnzeigen2()
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeNumbers
    .Icon = msoIconTip
    .Heading = blTitel
    blText = "Es wurden " & Anzahl & " Kontakte verarbeitet." & vbCrLf & vbCrLf
    blText = blText & "Die Werte im Feld " & Chr(34) & "Kategorien" & Chr(34)
    blText = blText & " stehen nun im Feld " & Chr(34) & Feld(Wahl, 1) & Chr(34)
    blText = blText & " zur Verfügung."
    If Not oBlase.Checkboxes(1).Checked Then
      blText = blText & vbCrLf & vbCrLf & "Der dynamisch erstellte "
      blText = blText & "Outlook-Ordner heisst " & Chr(34) & Ordner & Chr(34) & "."
    End If
    .Text = blText
    .Labels.Count = 0
    .Checkboxes.Count = 0
    .Button = msoButtonSetOK
    .Show
  End With
End Function
```

(Kopiert das Outlook-Feld "Kategorien" um. Die Wahl des Zielfeldes liegt beim Benutzer.

Seriendruck aus Outlook2000 statt aus MS Word initialisieren

Eingangs sei nochmals erwähnt, dass sich diese Lösungsvariante auf **Outlook2000** beschränkt.

Lösungsweg

Zusammenfassend sollten Sie bei diesem Ansatz folgende Schritte nachvollziehen:

1. Sie definieren in Outlook2000 eine Ansicht, welche nur die Kontakte einer Kategorie oder mehreren von Ihnen bestimmtem Kategorien enthält.
2. Sie initialisieren den Seriendruck aus Outlook2000 statt aus Word2000. Dabei werden an Word nur die Kontakte der aktuellen Ansicht übergeben.

Vorgehensweise
Folgende Schritt-für-Schritt-Anweisung geht davon aus, dass die Serienbrieffunktion nur jene Kontakte verarbeiten soll, welche im Feld *Kategorien* den Wert **Privat** aufweisen.

1. Starten Sie **Outlook2000** und wechseln Sie zum **Kontakte-Ordner**.
2. Vom Hauptmenü in Outlook2000 wählen Sie den Dialog ***Ansicht - Aktuelle Ansicht - Ansicht anpassen***.
3. Im erscheinenden Dialog wählen Sie die Schaltfläche ***Filtern...*** und wechseln in den Reiter **Erweitert**.
4. Klicken Sie auf die Schaltfläche ***Feld*** und wählen Sie dann ***Häufig verwendete Felder - Kategorien***.
5. Stellen Sie im Kombinationsfeld **Bedingung:** den Wert **ist (genau)** ein.
6. Schreiben Sie in das Feld **Wert:** den Wert **Privat**.
7. Schliessen Sie den Dialog mit ***Ok*** (2x) ab. Es werden Ihnen nun nur noch die Kontakte der Kategorie "Privat" angezeigt.
8. Wählen Sie nun vom Hauptmenü in **Outlook2000** den Dialog ***Extras - Seriendruck....***
9. Betätigen Sie nun die Schaltfläche ***Durchsuchen...*** und wählen Sie im nachfolgenden Dialog das Word-Seriendruckdokument aus.
10. Wenn Sie Ihre Aktion nun im Outlook-Dialog nochmals bestätigen, startet Word mit dem von Ihnen gewählten Seriendruckhauptdokument auf. Gleichzeitig werden Sie vom Office-Assistenten gefragt, ob Sie die Daten nun mit dem Hauptdokument zusammenführen möchten.
11. Nehmen Sie den Vorschlag des Assistenten an und geben Sie im folgenden Dialog noch an, ob das Resultat in ein neues Dokument geschickt werden soll oder aber direkt zum Drucker. Word verarbeitet nun nur diese Datensätze, welche Sie zuvor nach Kategorien gefiltert haben.

Links zu diesem Thema

Microsoft

Deutsch

► **WD97: Mit Seriendruck nach Kategorie sortierte Listen erstellen.**
► **OL97D: Ansicht filtern nach Geburtstagen / Feiertagen**
► **OL98: Programmatisches Filtern nach Kategorien**

Englisch

► **OL97: Categories Unavailable in Mail Merge to Word**
► **How to: Filter items in Outlook 97 by categories.**
► **OL98: How to Create Custom Views**

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/OLKateg.htm

René Probst, Oktober 1999

Änderungen
- April 2000 - Mit Outlook2000 ist es nun möglich die Serienbrieffunktion ex Outlook zu initialisieren.

Wie setze ich in einem Serienbrief eine bedingte Anrede?

Wie kann ich beim Seriendruck - Adressetiketten ein Bedingungsfeld einbringen?

Wie kann ich Leerschläge, welche aus leeren Felder resultieren, eliminieren?

◄ Zurück zum Hauptindex

Frage

> 1. Ich möchte einen Serienbrief fabrizieren. Der Text beinhaltet auch die Begrüssungsfloskel *Sehr geehrte(r)....* Wie kriege ich das am einfachsten hin, dass diese Anrede geschlechtsspezifisch ist?
>
> 2. Wenn ich in Word den Dialog *Seriendruck - Adressetiketten...* aufrufe und dann die Datenquelle öffne erhalte ich eine Maske, welche mir zwar erlaubt, Seriendruckfelder einzufügen, allerdings ist der Knopf mit der Bezeichnung Bedingungsfeld einfügen nicht erreichbar.
>
> 3. In meinem Serienbriefhauptdokument stehen die beiden Felder mit der Bezeichnung *Nachname* und *Vorname* nebeneinander, nur getrennt durch einen Leerschlag. bei einem Datensatz, bei dem das Feld Vorname leer ist, resultieren dann aber beim zusammenführen zwei Leerschläge daraus. Wie kann dieser zusätzliche und störende Leerschlag eliminiert werden?

Antworten

1. Frage

Sie erreichen dies mit einem **Bedingungsfeld** oder allenfalls mit **verschachtelten Bedingungsfeldern**, falls Sie auf mehr als zwei Alternative zurückgreifen möchten (*Sehr geehrte Frau, Sehr geehrter Herr, Sehr geehrte Damen und Herren*). Bei Bedingungsfeldern handelt es sich letztlich um Word-Feldfunktionen.

Wie Sie im Einzelnen vorgehen sollten, erfahren Sie im nun folgenden Text. Dabei werden immer **zwei Lösungsvarianten** vorgestellt. Die erste ist dialoggeführt (mittels Word-Dialoge) und die zweite Variante erklärt, wie Sie den Feldcode direkt in das Dokument eingeben können. Beide Varianten führen zu einem identischen Ziel und bedienen sich letztlich auch der gleichen Mitteln. Die zweite Methode ist eher für versierte Word-Benutzer gedacht.

Vor allem das Verschachteln von Feldfunktionen ist etwas kniffelig aber leider immer dann nötig, wenn Sie es bei Schwarz und Weiss nicht bewenden lassen wollen, sondern vielleicht auch Grau möchten. Dieses Skript geht deshalb auch speziell auf diese Verschachtelung ein.

2. Frage

Die Schaltfläche Bedingungsfeld einfügen ist wenn die Maske mit der Bezeichnung Etiketten erstellen angezeigt wird, tatsächlich nicht erreichbar, Sie können aber den Feldcode für das Bedingungsfeld manuell in diese Maske eintippen. Wie dies gemacht wird, entnehmen Sie bitte dem folgenden Text in diesem Skript.

Die beiden entsprechenden Kapitel sind wie folgt überschrieben: Feldfunktion direkt in das Dokument oder in die Etikettenmaske eingeben.

3. Frage

Auch für diesen Fall sollten Sie ein **Bedingungsfeld** einsetzten, welches bewirkt, dass der Feldwert nur dann eingesetzt wird, falls das entsprechende Feld für einen bestimmten Datensatz einen Wert aufweist.

Zwar verfüge der Seriendruck über die Option Aus Leerfeldern resultierende Leerzeilen nicht drucken, diese Option kommt aber nur dann zum tragen (wie der Name der Funktion auch suggeriert), wenn auf Grund eines leeren Feldinhaltes eine ganze Leerzeile entstehen würde. Die genannte Option finden Sie übrigens im Folgedialog, wenn Sie auf der Symbolleiste mit der Bezeichnung *Seriendruck die Schaltfläche Seriendruck...* anklicken.

Wie Sie in das Seriendruckhauptdokument ein Bedingungsfeld einfügen, welches diesen überflüssigen und störenden Leerschlag verhindert, erfahren Sie im entsprechenden Kapitel gegen Ende dieses Skriptes.

Vorarbeiten

Um die Aufgabenstellung zu üben, richten Sie bitte folgendes Umfeld ein:

1. Erstellen Sie eine neue Excel-Arbeitsmappe.
2. Unsere Tabelle soll drei Spalten haben mit folgenden Spaltenüberschriften:
   - **Name**
   - **Vorname**
   - **Sex**

   Geben Sie also bitte diese Überschriften in der ersten Zeile der Excel-Tabelle ein.
3. Tragen Sie ca. ein duzend Namen (also Datensätze) ein. Berücksichtigen Sie dabei weibliche, männliche und juristische Personen.
4. Tragen Sie für diese Datensätze in der Spalte Sex für weibliche Personen die Kennzeichnung **f**, für männliche Personen die Kennzeichnung **m** und für juristische Personen die Kennzeichnung **n** ein.
5. Speichern Sie nun die Mappe und verlassen Sie Excel.
6. Erstellen Sie in Word ein neues Dokument.
7. Wählen Sie vom Word-Hauptmenü den Dialog ***Extras - Seriendruck... - Erstellen - Serienbriefe... - Aktives Fenster***.
8. Wählen Sie nun immer noch in diesem **Seriendruckmanager** zuerst den Knopf **Datenquelle importieren** und wählen sie dann **Datenquelle öffnen**.
9. Achten Sie darauf, dass als Dateityp: *Excel-Arbeitsblätter* aktiviert ist und öffnen Sie die soeben erstellte Arbeitsmappe.
10. Bestätigen Sie diese Wahl nun zweimal und klicken Sie dann auf den Schaltknopf mit der Bezeichnung **Hauptdokument bearbeiten**.
11. In der kontextbezogenen Symbolleiste, klicken Sie nun auf die Schaltfläche mit der Bezeichnung **Seriendruckfeld einfügen** und fügen dann an geeigneter Stelle die Felder *Name* und *Vorname* in das Dokument ein.
12. Damit haben Sie ein Serienbriefhauptdokument eingereichtet "*and we are now ready for take-off!*"

Bedingungsfeld für zwei Bedingungen einrichten

Mittels Word-Dialog

Wenn Sie nur zwischen zwei Anreden unterscheiden möchten ist die Sache einfach zu bewerkstelligen. Gehen Sie in diesem Falle wie folgt vor:

1. Plazieren Sie die Einfügemarke an die gewünschte Stelle im Dokument und wählen Sie aus dem kontextbezogenen Menü den Schaltknopf **Bedingungsfeld einfügen**.
2. Im aufgeklappten Menü wählen Sie nun den Eintrag **Wenn...Dann...Sonst...**.
3. Stellen Sie bei *Feldname*: den Wert **Sex** ein und tippen Sie in das Textfeld mit der Bezeichnung *Vergleichen mit*: den Buchstaben **m** ein.
4. Schreiben Sie nun in das Feld mit der Bezeichnung *Dann diesen Text einfügen*: den Satz **Sehr geehrter Herr** und in das Feld mit der Bezeichnung *Sonst diesen Text einfügen*: den Satz **Sehr geehrte Frau**. Schliessen Sie dann den Dialog mit **Ok** ab.

Das wäre dann auch schon alles. Um das Ergebnis Ihrer Bemühungen zu betrachten, klicken Sie auf der kontextbezogenen Symbolleiste auf die Sachaltfläche **ABC**. Mit den Pfeilsymbolen auf der selben Leiste können Sie sich zwischen den Datensätzen bewegen und das Resultat bewerten.

Feldfunktion direkt in das Dokument oder in die Etikettenmaske eingeben

Sie können die Feldfunktion, welche aus dem Vorgehen resultiert, welches so eben beschrieben wurde auch direkt in das Dokument eingeben. Diese Feldfunktion sieht dann wie folgt aus:

Word 97

```
{ WENN { Seriendruckfeld Sex } = "m" "Sehr geehrter Herr"
"Sehr geehrte Frau" }
```

Word2000

```
{ IF { MERGEFIELD Sex } = "m" "Sehr geehrter Herr" "Sehr ge-
ehrte Frau" }
```

Die geschweiften Klammern repräsentieren Start und Ende von Feldfunktionen. Wie Sie eine Feldfunktion direkt in das Dokument einfügen, erfahren Sie in einem speziellen Kapitel etwas weiter unten in diesem Skript.

Bedingungsfeld für mehr als zwei Bedingungen einrichten

Mittels Word-Dialog

Wenn sie zwischen drei (oder sogar mehr) Anreden unterscheiden möchten, wird das Vorgehen etwas kniffliger. Im nachfolgenden wird erklärt, wie Sie vorgehen sollten, um neben weiblichen und männlichen Personen auch juristische Personen einzubeziehen. Die Beschreibung deckt also nicht nur zwei sondern drei Fälle ab.

1. Plazieren Sie die Einfügemarke an die gewünschte Stelle im Dokument und wählen Sie aus dem kontextbezogenen Menü den Schaltknopf **Bedingungsfeld einfügen**.
2. Im aufgeklappten Menü wählen Sie nun den Eintrag **Wenn...Dann...Sonst...**.
3. Stellen Sie bei *Feldname:* den Wert **Sex** ein und tippen Sie in das Textfeld mit der Bezeichnung *Vergleichen mit*: den Buchstaben **m** ein.
4. Schreiben Sie nun in das Feld mit der Bezeichnung *Dann diesen Text einfügen*: den Satz **Sehr geehrter Herr**. Das Feld mit der Bezeichnung *Sonst diesen Text einfügen*: lassen Sie leer und schliessen den Dialog mit **Ok** ab.
5. Drücken Sie nun die Tastenkombination **[Alt + F9]**. Damit wird der **Feldcode** des Bedingungsfeldes, welches Sie soeben erstellt haben, angezeigt .
6. Platzieren Sie nun die Einfügemarke **zwischen das letzte Paar der Gänsefüsschen** im Feldcode. Zwischen diesem Paar Gänsefüsschen ist noch kein Text, sie folgen unmittelbar aufeinander.
7. Klicken Sie nun abermals auf den Schaltknopf **Bedingungsfeld einfügen**.
8. Im aufgeklappten Menü wählen Sie nun den Eintrag **Wenn...Dann...Sonst...**.
9. Stellen Sie bei *Feldname*: den Wert **Sex** ein und tippen Sie in das Textfeld mit der Bezeichnung *Vergleichen mit*: den Buchstaben **f** ein.
10. Schreiben Sie nun in das Feld mit der Bezeichnung *Dann diesen Text einfügen*: den Satz **Sehr geehrte Frau** und in das Feld mit der Bezeichnung *Sonst diesen Text einfügen*: den Satz **Sehr geehrte Damen und Herren**. Schliessen Sie dann den Dialog mit **Ok** ab.
11. Drücken Sie nun abermals die Tastenkombination **[Alt + F9]**, um den Feldcode wieder auszublenden.

Damit haben Sie ein **verschachteltes Bedingungsfeld** eingerichtet.. Um das Ergebnis Ihrer Bemühungen zu betrachten, klicken Sie in der kontextbezogenen Symbolleiste auf die Sachaltfläche **ABC**. Mit den Pfeilsymbolen auf der selben Symbolleiste können Sie sich zwischen den Datensätzen bewegen und das Resultat bewerten.

Sollen noch mehr als drei Varianten abgedeckt werden, wiederholen Sie den beschriebenen Vorgang in analoger Weise. Spätestens dann werden Sie wissen, was hier unter dem Begriff *verschachteln* gemeint ist.

Feldfunktion direkt in das Dokument oder in die Etikettenmaske eingeben

Sie können die Feldfunktion, welche aus dem Vorgehen resultiert, welches so eben beschrieben wurde auch direkt in das Dokument eingeben. Diese Feldfunktion sieht wie folgt aus:

Word 97

```
{ WENN { Seriendruckfeld Sex } = "m" "Sehr geehrter Herr" "{ WENN
{ Seriendruckfeld Sex } = "f" "Sehr geehrte Frau" "Sehr geehrte
Damen und Herren" }" }
```

Word2000

```
{ IF { MERGEFIELD Sex } = "m" "Sehr geehrter Herr" "{ IF { MERGE-
FIELD Sex } = "f" "Sehr geehrte Frau" "Sehr geehrte Damen und
Herren" }" }
```

Die geschweiften Klammern repräsentieren Anfang und Ende von Feldfunktionen. Wie Sie eine Feldfunktion direkt in das Dokument einfügen, erfahren Sie in einem speziellen Kapitel etwas weiter unten im Skript.

Anmerkung

- MS Word versteht nur diese, etwas unleserliche Form, eine Mehrfachbedingung zu codieren.

Aus Leerfeldern resultierende Leerschläge vermeiden

Wie schon Eingangs erwähnt, ist auch hier die Verwendung eines **Bedingungsfeldes** angesagt. Auch hier gilt, Sie können ein solches Bedingungsfeld über eine Dialog eingeben, oder aber Sie schreiben die Feldfunktion, welche dazu nötig ist direkt in Ihr Dokument.

Angenommen Sie verfügen über einen Datenbestand, welcher sich dadurch auszeichnet, dass das Feld mit der Bezeichnung *Vorname* in einzelnen Datensätzen leer ist und Sie wollen vermeiden, dass dadurch einen überschüssigen Leerschlag resultiert, dann gehen Sie wie folgt vor:

Mittels Word-Dialog

1. Plazieren Sie die Einfügemarke an die gewünschte Stelle im Dokument und wählen Sie aus dem kontextbezogenen Menü den Schaltknopf **Bedingungsfeld einfügen**.
2. Im aufgeklappten Menü wählen Sie nun den Eintrag **Wenn...Dann...Sonst...**.
3. Stellen Sie im Kombinationsfeld mit der Bezeichnung *Feldname*: den Wert **Vorname** ein und im Kombinationsfeld mit der Bezeichnung *Vergleich: den Wert* **Ist nicht leer** ein.
4. Das Feld mit der Bezeichnung *Vergleichen mit*: lassen Sie gefälligst *leer*..
5. Platzieren Sie nun die Einfügemarke in das Feld mit der Bezeichnung **Dann diesen Text einfügen**:
6. Drücken Sie nun die Tastenkombination **[Strg + F9]**. Damit wird ein **Paar von geschweiften Klammern** erzeugt.
7. Platzieren Sie nun die Einfügemarke genau zwischen diese geschweiften Klammern und tippen Sie folgendes ein `Seriendruckfeld Vornamen` **Word 97** bzw.
   `Mergefield Vorname` **Word2000**.

   Als ganzes sieht dies nun also wie folg aus
   { `Seriendruckfeld Vornamen` } **Word 97** bzw. { `Mergefield Vornamen` } **Word2000**.
8. Das Feld mit der Bezeichnung *Sonst diesen Text einfügen*: lassen Sie gefälligst ebenfalls *leer*..
9. Schliessen Sie dann den Dialog mit **Ok** ab.

Feldfunktion direkt in das Dokument oder in die Etikettenmaske eingeben

Sie können die Feldfunktion, welche aus dem Vorgehen resultiert, welches so eben beschrieben wurde auch direkt in das Dokument eingeben. Diese Feldfunktion sieht wie folgt aus:

Word 97

```
{ IF { Seriendruckfeld Vorname }<> "" "{ Seriendruckfeld Vorname }" "" }
```

Word2000

```
{ IF { MERGEFIELD Vorname }<> "" "{ MERGEFIELD Vorname }" "" }
```

Die geschweiften Klammern repräsentieren Anfang und Ende von Feldfunktionen. Wie Sie eine Feldfunktion direkt in das Dokument einfügen, erfahren Sie gleich anschliessend.

Wie gibt man eine Feldfunktion in MS Word ein?

Sollten es Sie reizen, die Feldfunktion direkt, also ohne Dialog zu erstellen, dann ist im folgenden beschrieben, wie man eine Feldfunktion in ein Word-Dokument einbringt.

1. Betätigen Sie die Tastenkombination **[Alt + F9]** um die Feldcodes einzublenden.
2. Generieren Sie das Gerüst für den neuen Feldcode, indem Sie die Tastenkombination **[Strg + F9]** betätigen. Sie sollten nun ein Pärchen geschweifter Klammern sehen; die Einfügemarke steht mitten drin.
3. Schreiben Sie nun die Abfrage, wie oben abgebildet.
4. Betätigen Sie nun wieder die Tastenkombination [**Alt + F9]** um den Feldcode auszublenden und in die Normalansicht der Felder zurückzukehren.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/BFeld.htm

René Probst, November 2000

Änderungen

- 12. März 2011 – Lerstellen unterdrücken

Der Text in konventionellen Formulartextfeldern wird abgeschnitten oder verschluckt

◄ Zurück zum Hauptindex

Symptome

> Sie geben in ein konventionelles Formulartextfeld Text ein, während das Formular geschützt ist. Sowie Sie das Feld verlassen, ist der Text verschwunden oder es werden nur die ersten vier Zeichen angezeigt.

Erklärung

Beim Text den Sie eingetippt haben, handelt es sich mit höchster Wahrscheinlichkeit um ein Kandidat für die AutoKorrektur.

In MS Word gilt: Wenn das erste Wort eines Formularfeldes ein Kandidat für die AutoKorrektur ist, und die Länge des Feldes auf unbegrenzt gesetzt ist, kommt es zum erwähnten Fehlverhalten. Der Fehler wird weder durch einen der Service Releases von Office noch mit Word2000 behoben.

Lösung

Setzen sie die Länge von Textformularfeldern zwar grosszügig, vermeiden Sie aber eine unbegrenzte Länge. Und so setzen Sie eine feste Länge für ein Formularfeld: (Dies betrifft Felder mit dem Format **Normaler Text**).

1. Wählen Sie vom Word-Hauptmenü den Dialog ***Extras - Dokumentschutz aufheben***.
2. Führen Sie einen Rechtsklick auf das Feld aus und wählen Sie im Kontextmenü ***Eigenschaften***.
3. Geben Sie im Feld **Maximale Länge** eine feste Länge ein.

VBA-Code

Testformularfeldern automatisiert eine feste Länge zuzuweisen

Bei vielen Feldern oder vielen Formularen mag es sinnvoll erscheinen, diesen Vorgang zu automatisieren. Nachfolgend ein Makro, welches dieses für alle Formulartextfelder innerhalb einer Dokumentenvorlage oder eines Dokumentes erledigt. Im Code-Beispiel, wird eine Länge von 80 Zeichen verwendet.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub FormularfeldFesteLängeSetzen()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Dim x As FormField
  If oDoc.ProtectionType > 0 Then oDoc.Unprotect
  For Each x In oDoc.FormFields
    If x.Type = wdFieldFormTextInput Then
      If x.TextInput.Type = wdRegularText Then x.TextInput.Width = 80
    End If
  Next x
End Sub
```

(Alle Textformularfelder mit dem Format "Normaler Text" erhalten eine feste Länge verpasst.)

Links zu diesem Thema

Microsoft

► **WD97: AutoCorrect Text Deleted, Partially Visible in Form Field**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/FFA.htm

René Probst, Oktober 1999

Änderungen

- April 2000 - VBA-Code korrigiert

Der Befehl Suchen/Ersetzen funktioniert nicht in Kopf- und Fusszeilen und auch nicht in Textfeldern

Wie kann ich die Suchen/Ersetzen-Funktion auf ein ganzes Verzeichnis mit Word-Dokumenten wirken lassen?

◄ Zurück zum Hauptindex

Problem

> 1. Wende ich den Befehl Suchen/Ersetzen an, wird dieser Befehl zwar korrekt im Hauptteil des Dokumentes ausgeführt, sowohl Kopf- und Fusszeilen wie auch Textfelder werden davon aber nicht erfasst.
>
> 2. Ich suche eine Routine, welche es mir ermöglicht, die Suchen/Ersetzen-Funktion von MS Word auf ein ganzes Verzeichnis mit Word-Dokumenten wirken zu lasen.

Erklärung

Die Beobachtung, dass sich die Suchen/Ersetzen-Funktion ausschliesslich auf den Hauptteil des Dokumentes auswirkt, ist richtig. Das Word-Objekt-Modell kennt nämlich nicht weniger als elf Dokumententeile. Diese Teile werden in der Fachsprache **Story Ranges** genannt. Allein sechs dieser *Story Ranges* betreffen Fuss- und Kopfzeilen.

Lösung

Um die Suchen/Ersetzen-Funktion in diesen Dokumententeile erfolgreich durchzuführen sollten Sie zuerst den fraglichen Teil aktivieren (also z.B. in die Kopfzeilenansicht wechseln). Alternativ können Sie aber auch eines der hier publizierten Makros verwenden..

VBA-Code

Makro um eine Zeichenfolge in allen Teilen eines Dokumentes («*Story Ranges*») zu ersetzen

Bevor Sie beginnen...
Passen Sie diesen Code Ihren Bedürfnissen an, bevor sie los legen. Das ist nicht schwierig, weil es nur einer einfachen Anpassung bezüglich der Code-Zeilen im Kopf des Makros bedarf. Gehen Sie wie folgt vor:

- Geben Sie für die Konstante **Suche** diejenige Zeichenfolge ein, nach der gesucht werden soll.
- Setzen Sie für die Konstante **ErsetzeMit** diejenige Zeichenfolge mit der die zu suchende Zeichenfolge ersetzt werden soll.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Suche = "F. Muster"
Private Const ErsetzeMit = "Felix Muster"
Private Teil As Range

Sub SuchenErsetzenSchleife()
  Application.ScreenUpdating = False
  For Each Teil In ActiveDocument.StoryRanges
    SuchenErsetzen
    While Not (Teil.NextStoryRange Is Nothing)
      Set Teil = Teil.NextStoryRange
      SuchenErsetzen
    Wend
  Next
  Application.ScreenUpdating = True
End Sub

Sub SuchenErsetzen()
  Teil.Find.Execute FindText:=Suche, _
    ReplaceWith:=ErsetzeMit, _
    Replace:=wdReplaceAll
End Sub
```

(Ersetzt einem im Kopf des Codes nominierten Text mit einem anderen und zwar in allen im Dokument verfügbaren Texteilen.)

Makro um eine Zeichenfolge in allen Teilen eines Dokumentes zu ersetzen (interaktive)

Hier folgt ein ähnliches Makro wie oben abgebildet. Dieses Mal werden Sie aber beim Aufruf des Makros zuerst nach der Zeichenfolge, nach der gesucht werden soll gefragt. Gleiches gilt auch für die Zeichenfolge, welche als Ersetzung dient. Da dieser Code Ihre Bedürfnisse interaktiv ermittelt, Bedarf es keiner Anpassung.

```
Private Const Titel = "Suchen/Ersetzen..."
Private Const M1 = "Welche Zeichenfolge soll ersetzt werden?"
Private Const M2 = "Welche Zeichenfolge soll neu gelten?"
Private Teil As Range
Private Suche As String
Private ErsetzeMit As String

Sub SuchenErsetzenSchleifeInteraktiv()
  If Selection.Type = 2 Then
    Sel = Selection.Text
    If Right(Sel, 1) = Chr(13) Then Sel = Left(Sel, Len(Sel) - 1)
  End If
  Antwort = InputBox(M1, Titel, Sel)
  If Antwort = "" Then End
  Suche = Antwort
  Antwort = InputBox(M2, Titel)
  If Antwort = "" Then End
  ErsetzeMit = Antwort
  Application.ScreenUpdating = False
  For Each Teil In ActiveDocument.StoryRanges
    SuchenErsetzen
    While Not (Teil.NextStoryRange Is Nothing)
      Set Teil = Teil.NextStoryRange
      SuchenErsetzen
    Wend
  Next
  Application.ScreenUpdating = True
End Sub

Sub SuchenErsetzen()
  Teil.Find.Execute FindText:=Suche, _
    ReplaceWith:=ErsetzeMit, _
    Replace:=wdReplaceAll
End Sub
```

(Fragt nach einem Text und ersetzt diesen mit einem anderen Text, welcher ebenfalls beim Benutzer angefragt wird. Das Ersetzen geschieht in allen im Dokument verfügbaren Texteilen.)

☹ Wegen eines Fehlverhaltens von MS Word, wird der Text von Textfeldern, wenn sich letztere in der Kopf- oder Fusszeile befinden, nicht erfasst. Siehe dazu den Link zur Datenbank von Microsoft am Ende dieses Skriptes.

Makro um eine Zeichenfolge in allen Dokumenten, welche sich im nominierten Verzeichnis befinden zu ersetzen. Der Code berücksichtigt alle Teile der Dokumente (Story Ranges).

Bevor Sie beginnen...
Passen Sie diesen Code Ihren Bedürfnissen an, bevor sie los legen. Das ist nicht schwierig, weil es nur einer einfachen Anpassung bezüglich der Code-Zeilen im Kopf des Makros bedarf. Gehen Sie wie folgt vor:

- Setzen Sie den Namen des Verzeichnisses, welches durchsucht werden soll in der Konstante **Verzeichnis**.
- Geben Sie in der Konstanten **Filter** eine Maske ein, welche die zu durchsuchenden Dateien erfüllen müssen. Wollen Sie alle Word-Dokumente durchsuchen, lautet der Wert ***.doc**. Möchten Sie zum Beispiel nur die Dokumente durchsuchen, deren Name mit einem **D** beginnt, setzen Sie den Filter auf den Wert **D*.doc**.
- Möchten Sie neben dem Verzeichnis, welches Sie angegeben haben auch deren Unterverzeichnisse durchsuchen, setzen Sie den Wert der Konstante **UnterverzeichnisseDurchsuchen** auf den Wert **1**, möchten Sie dies nicht lautet der Wert 0. Setzen Sie diesen numerischen Wert **nicht** in Gänsefüsschen.
- Geben Sie für die Konstante **Suche** diejenige Zeichenfolge ein, nach der gesucht werden soll.
- Setzen Sie für die Konstante **ErsetzeMit** diejenige Zeichenfolge mit der die zu suchende Zeichenfolge ersetzt werden soll.

```
' **** Anpassbare Werte ****
Private Const Verzeichnis = "C:\Eigene Dateien"
Private Const Filter = "*.doc"
Private Const UnterverzeichnisseDurchsuchen = 0
Private Const Suche = "Felix Muster"
Private Const ErsetzeMit = "Felix Muster"
' **** Ende der Anpassung ****

Private Teil As Range

Sub SuchenErsetzenGanzesVerzeichnis()
  Dim oDoc As Document
  tmp = UnterverzeichnisseDurchsuchen
  If tmp = 1 Then UVD = True Else UVD = False
  If Documents.Count > 0 Then Dokument = ActiveDocument.FullName
  With Application.FileSearch
    .LookIn = Verzeichnis
    .FileName = Filter
    .SearchSubFolders = UVD
    .Execute SortBy:=msoSortByFileName
    Anzahl = .FoundFiles.Count
    Application.ScreenUpdating = False
    For Each aDok In .FoundFiles
      If aDok <> Dokument Then
        On Error Resume Next
        Documents.Open aDok
        Fehler = Err.Number
        On Error GoTo 0
        If Fehler = 0 Then
          Set oDoc = ActiveDocument
```

```
          If oDoc.ProtectionType = wdNoProtection Then
            If oDoc.ReadOnly = False Then
              StatusBar = "Durchsuche Dokument " + aDok + "."
              DoEvents
              SuchenErsetzenSchleife
              oDoc.Close SaveChanges:=wdSaveChanges
            Else
              oDoc.Close SaveChanges:=wdDoNotSaveChanges
            End If
            Else
            oDoc.Close SaveChanges:=wdDoNotSaveChanges
          End If
        End If
      End If
    Next
  End With
  StatusBar = CStr(Anzahl) + " Dokumente durchsucht."
  DoEvents
  Application.ScreenUpdating = True
End Sub

Private Sub SuchenErsetzenSchleife()
  Application.ScreenUpdating = False
  For Each Teil In ActiveDocument.StoryRanges
    SuchenErsetzen
    While Not (Teil.NextStoryRange Is Nothing)
      Set Teil = Teil.NextStoryRange
      SuchenErsetzen
    Wend
  Next
End Sub

Private Sub SuchenErsetzen()
  Teil.Find.Execute FindText:=Suche, _
    ReplaceWith:=ErsetzeMit, _
    MatchCase:=GrossUndKleinSchreibung, _
    MatchWholeWord:=GanzesWort, _
    MatchWildcards:=Jocker, _
    Replace:=wdReplaceAll
End Sub
```

(Ersetzt eine nominierte Zeichenfolge in allen Dokumenten eines Verzeichnisses. Der Code berücksichtigt alle Dokumententeile (Story Ranges) eines Dokumentes.)

Anmerkungen

- Geschützte Dokumente werden nicht verarbeitet.
- Dokumente, welche Automationsroutinen enthalten (AutoOpen etc.) können Probleme verursachen.
- Wegen eines Fehlverhaltens von MS Word, wird der Text von Textfeldern, wenn sich letztere in der Kopf- oder Fusszeile befinden, nicht erfasst. Siehe dazu den Link zur Datenbank von Microsoft am Ende dieses Skriptes.

Die elf Gut-Nachtgeschichten von Word

Hier zur Illustration noch, die elf Dokumententeile, welche im Objektmodell von Word definiert sind und folglich durchsucht werden müssen:

Komponente	Word-Objekt	Relation	Bemerkung
Haupttext	MainTextStory	1 pro Dokument	MainTextStory und Dokument können synonym behandelt werden.
Hauptkopfzeile	PrimaryHeaderStory	1 pro Abschnitt (Section)	Ist immer vorhanden, kann aber auch leer sein. Kann auch der OddPageHeaderStory entsprechen, welche es nominell nicht gibt
Kopfzeile für erste Seite	FirstPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Erste Seite anders" aktiviert ist.
Kopfzeile für gerade Seite	EvenPageHeaderStory	0 oder 1 pro Abschnitt (Section)	Nur vorhanden, wenn im Seitenlayout "Gerade-/ungerade Seiten anders" aktiviert ist.
Hauptfusszeile	PrimaryFooterStory	1 pro Abschnitt (Section)	siehe Hauptkopfzeile
Fusszeile für erste Seite	FirstPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben
Fusszeile für gerade Seite	EvenPageFooterStory	0 oder 1 pro Abschnitt (Section)	siehe oben
Kommentare	CommentsStory	0 zu n pro Dokument	Kommentare können nur Objekte (Elemente) der MainTextStory sein
Endnote	EndnotesStory	0 oder 1 pro Dokument	Endnoten können nur Objekte (Elemente) der MainTextStory sein
Fussnoten	FootnotesStory	0 zu n pro Dokument	Fussnoten können nur Objekte (Elemente) der MainTextStory sein
Textfeld	TextFrameStory	0 zu n pro Story	Textfelder sind in erster Linie Objekte (Elemente) von Shapes

Links zu diesem Thema

Microsoft

► **WD2000: StoryRanges Does Not Return Text Box Text in Header/Footer**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SE.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich eine Tabelle ohne Gitter erzeugen bzw. eine solche zum Standard küren?

◄ Zurück zum Hauptindex

Problem

> **Wenn ich über den Dialog *Tabelle - Zellen einfügen - Tabelle - AutoFormat...* das Format *(ohne)* wähle, dann wird mir keine Tabelle ohne Gitterlinien eingefügt. Vielmehr hat dies den selben Effekt, als würde ich das Format *Einfach 1* auswählen. Dadurch ist es mir auch nicht möglich, eine Tabelle ohne Gitterlinien zum Standard zu küren.**

Erklärung

Es handelt sich dabei um ein Fehler in Word, welcher man sowohl in Word 97 wie auch in Word2000 findet. Allerdings bietet Word2000 eine elegantere Umgehungsmöglichkeit.

Abhilfe

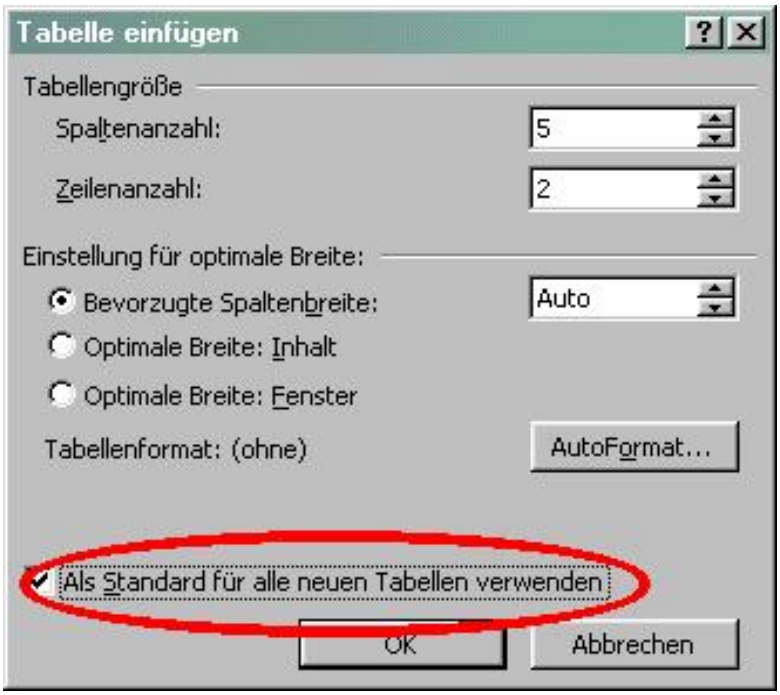
Ich kenne drei Umgehungsmöglichkeiten für dieses Problem.

- Wenn sich die Einfügemarke in der Tabelle befindet können Sie die Tastenkombination **[Strg + Alt + U]** betätigen. Die Gitterlinien werden nun in nicht druckbare Linien konvertiert.
- Eine weitere Möglichkeit wäre, alle formatbezogenen Häkchen im Dialog ***Tabelle - Tabelle AutoFormat...*** zu entfernen.
- Sie können auch das unten publizierte Makro verwenden.

Word2000

Word2000 bietet die Option ein bestimmtes Tabellenformat zum Standard zu küren. Gehen Sie wie folgt vor:

1. Wählen Sie den Dialog Tabelle - Zellen einfügen - Tabelle.
2. Setzen Sie das Häkchen bei Als Standard für alle neuen Tabellen verwenden.
3. Betätigen Sie die Schaltfläche ***AutoFormat...***.
4. Wählen Sie in der Liste das Format **Einfach 1**, also **nicht** *(Ohne)*.
5. Schalten Sie alle Formatierungsoptionen aus und schliessen Sie den Vorgang mit ***2x Ok*** ab.



VBA-Makro

```
Sub TabelleEinfügenOhne()
  With Dialogs(wdDialogTableInsertTable)
    Antwort = .Show
    tFormat = .Format
  End With
  If Antwort = 0 Then End
  If tFormat = wdTableFormatNone Then
    Selection.Tables(1).AutoFormat Format:=wdTableFormatNone
  End If
End Sub
```

(Fügt eine Tabelle ohne Gitter ein.)

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/TaboG.htm

René Probst, Oktober 1999

Änderungen

- 9. November 2000 - Screen shot eingefügt
- 14. November 2000 - Beschreibung war schlecht

Wieso kann ich nicht auf das Adressbuch von Outlook Express 4 oder 5 zugreifen.

◄ Zurück zum Hauptindex

Symptome

> **Ich habe unter Word den Befehl *Einfügen - Adresse* eingerichtet. Wenn ich diesen Dialog aufrufe, wird mir eine Liste von möglichen Adressbüchern angezeigt (z.B. Kontakte, Adressbuch, Outlook Adressbuch, Schedule+). Den Eintrag Outlook Express kann ich aber nicht finden. Wähle ich den Eintrag Outlook Adressbuch, dann stehen entweder keine oder die falschen Adressen zur Verfügung.**

Ursache

Word und andere Office-Anwendungen können **nicht** auf das Adressbuch von Outlook Express 4 oder Outlook 5 zugreifen. Dazu ist ein Adressbuch nötig, welches dem **MAPI**-Standard folgt. Das Format des Adressbuches von Outlook Express ist aber proprietär (keinem Standard folgend).

Rundherumläufe

1. Sie können das Adressbuch von Outlook Express exportieren und zwar in das Persönliche Adressbuch für Microsoft Exchange - Exchange Personal Address Book - (PAB).
2. Sie können von Outlook 97/98 oder 2000 das Adressbuch von Outlook Express in den **Kontakte-Ordner importieren**.

Export des Outlook Express Adressbuches

Um dieses Adressbuch zu exportieren, brauchen Sie die Komponente **Microsoft Exchange**. Dieses Produkt schafft aber zumindest im Zusammenhang mit **Windows 98** sehr viel Ärger. Dies beginnt bereits mit der Installation. Wenn Sie es dennoch versuchen möchten, finden Sie den Link zu dem entsprechenden Artikel von Microsoft weiter unten. Bitte beachten Sie, dass MS Exchange in **Windows 98** nicht mehr im Windows-Setup enthalten ist. Sie finden das Installationsprogramm, gut versteckt, im Verzeichnis **tools\oldwin95\Message**. Dort befindet sich auch eine grössere *Lies Mich-Datei*, welche gespickt ist mit wohlgemeinten Ratschlägen, wie Sie all die Klippen vielleicht doch noch erfolgreich umschiffen können. Bei der Installation oder Teilinstallation von MS Exchange werden übrigens Programmbibliotheken (DLL's) überschrieben, welche für andere Programme vital sein können. Gefährdet sind insbesondere Outlook 9x, Outlook Express und Netscape. Interessanterweise haben Sie es in diesem Zusammenhang etwas leichter, wenn Sie Windows 98 über Windows 95 installiert haben, als wenn Sie Windows 98 auf eine frisch formatierte Festplatte aufgespielt haben. Ich persönlich weigere mich, für ein dermassen fehleranfälliges Produkt Unterstützung, zu bieten. Letztlich bin ich auch hier für das Verursacherprinzip, was in diesem Zusammenhang nichts anderes heisst, dass die Leute welche den Mist produzieren, diesen gefälligst wieder wegräumen sollen. Kontaktieren Sie bei Problemen in diesem Umfeld also die Firma Microsoft.

Import des Outlook Express Adressbuches nach Outlook 97/98 oder 2000

Das Vorgehen beim Importieren des Outlook Express Adressbuches nach einer der MS Office Outlook-Versionen variiert je nach MS Outlook-Version, welche auf Ihrem System installiert ist. Hier stellvertretend eine Schritt für Schritt Anweisung bezogen auf **Outlook2000**. Benutzen Sie eine ältere Version von MS Outlook, dann besuchen Sie bitte den entsprechenden Link am Schluss dieses Kapitels.

1. Starten Sie Outlook2000.
2. Vom Hauptmenü wählen Sie den Punkt ***Datei - Importieren/Exportieren...***.
3. Wählen Sie im erscheinenden Dialog den Punkt **Internet-Mail und Adressen importieren** und fahren Sie mit ***Weiter*** fort.
4. In der nun folgenden Auflistung wählen Sie **Outlook Express 4.x, 5**.
5. Im unteren Teil des Dialoges aktivieren Sie nun die Option **Adressbuch importieren** und fahren dann mit ***Weiter*** fort.
6. Im nächsten Dialog wählen Sie als Ziel für die importierten Adressen **Outlook-Kontakte-Ordner** und bei den Optionen ist es wohl am sinnvollsten die Option **Keine Duplikate** zu verwenden.
7. Schliessen Sie den Dialog mit ***Fertigstellen*** ab.
8. Die Adressen werden Nun von Outlook-Express nach Outlook importiert.

Wichtig

! Verwechseln Sie Outlook (OL) und Outlook Express (OE) nicht miteinander. OL ist ein Bestandteil des ziemlich teuren Microsoft Office Paket, OE (welcher mir persönlich an sich gut gefällt) ist ein Bestandteil des Internet Explorers. Der Internet Explorer wiederum ist Bestandteil von Windows. Er wird auch gratis über die verschiedensten Kanäle vertrieben. Ungeschickt und nicht ganz ehrlich von Microsoft ist, dass *Internet Mail* nach *Outlook Express* umbenannt wurde, denn MS Outlook und Outlook Express sind technisch und lizenztechnisch gesehen, völlig verschiedene Produkte, wenn sie auch teilweise den selben Funktionsumfang abdecken.

! Mit dem Export der Adressen nach MS Outlook erhalten Sie eine **Momentaufnahme** der Adressen welche Sie unter Outlook Express administrieren. Dies allerdings im anerkannten MAPI-Format. Dieses Abbild wird aber nicht automatisch synchronisiert, wenn Sie Ihre Adresskarte weiterhin in Outlook Express nachführen. Wie auch immer, z. B. für Leute, welche nur sehr sporadisch unter Word einen Serienbrief erstellen möchten, und Outlook Express als Quelle verwenden möchten, ist das hier beschrieben Vorgehen ein durchaus sinnvoller Weg.

Links zu diesem Thema

Microsoft

► **How to Import and Export Address Books in Outlook Express**
► **OL97: How to Import Addresses from Outlook Express**
► **OL98: How to Import Addresses from Outlook Express**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/OE.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich die Adressfelder und deren Reihenfolge, welche ich über den Befehl Einfügen Adresse in mein Dokument einfüge beeinflussen?

◄ Zurück zum Hauptindex

Aufgabenstellung

> In Word gibt es einen praktischen Befehl, welcher mir erlaubt, Adressen aus dem persönlichen Adressbuch (PAB) oder aus dem Outlook 97/98/2000 Kontakte-Ordner zu übernehmen. Zusätzlich zu den Feldern, welche mir geliefert werden, brauche ich aber noch die Telefonnummer. Zudem möchte ich zwei Felder in ihrer Position verändern.

Lösung

Sie können sowohl die Felder, welche in der Adresse erscheinen sollen und die Anordnung der Felder weitgehend Ihren spezifischen Wünschen anpassen. Microsoft sieht dafür ein **AutoText**-Eintrag mit einem ganz bestimmten Namen vor. In der deutschen Ausgabe von Office lautet dieser Name:

- **AdreßLayout Word 97**
- **AdressLayout Word2000**

Für ein Gelingen, achten Sie bitte auf folgende Punkte:

! Achten Sie auf die hier abgedruckte Schreibweise des Namens für den AutoText-Eintrag!

- Bevor Sie damit beginnen, die Adresse neu zu gestalten, müssen Sie in Erfahrung bringen, unter welchen **Feldnamen** die einzelnen Felder erkannt werden. Die Tabelle, etwas weiter unten, gibt Ihnen darüber Auskunft.
- Alle Feldnamen müssen in **spitze Klammern** gesetzt werden.
- Beachten Sie weiter, dass ausschliesslich **Grossbuchstaben** zur Anwendung kommen.
- Sie dürfen **maximal zwölf Feldnamen** verwenden.
- Falls Sie mit einer einzigen Adressdefinition nicht über die Runde kommen, müssen Sie für jede Adressdefinition eine eigene Dokumentenvorlage erstellen. Falls Ihnen eine Definition genügt, ist es wohl sinnvoll, den AutoText-Einträge, welcher für das Adresslayout bestimmend ist, in der Standardvorlage **Normal.dot** abzulegen.
- Obwohl sowohl Outlook wie auch das PAB und Schedule^{+} MAPI-fähig sind, sind die Felder in einigen Fällen nicht deckungsgleich. Gewisse Felder fehlen in der einen oder andern Anwendung ganz. Word selber bringt eine zusätzliche Einschränkung ein. Word kennt nämlich nur etwa vierzig Felder. Nötig um das ganze Datenmodell in diesem Bereich abzudecken, wären aber rund achtzig Felder. Die Informatik lehrt im ersten Studiensemester, dass vor jedem grösserem EDV-Projekt die Datenmodulierung stehen sollte. Bei der Datenmodulierung geht es darum, die zu verarbeitenden Daten ganz eindeutig zu umschreiben und die Beziehungen zwischen den einzelnen Elementen zu eruieren. Die Leute, welche hier damit betraut wurden, scheinen dies leider nicht begriffen zu haben. Anders ist das Durcheinander dieser Felder und Bezeichnungen wohl nicht zu erklären. Erschwerend kommt noch hinzu, dass Begriffe wie z. B. *Titel* oder *Anrede* im angelsächsischen Sprachraum eine andere Bedeutung zu kommt, als bei uns. Sowohl die nachfolgende Tabelle wie auch der eingeblendeten Links zur *Knowledge Base* von Microsoft soll einen Überblick geben, welche Felder Sie aus welcher Anwendung unter welchem Feldnamen ansprechen können. Zu diesem Thema kursieren zahlreiche Skripts, welche sich leider in zum Teil widersprechen. Einige davon verbreiten also Fehlinformationen.

Tabelle der Adressfeldnamen

Feldname	Name wie im Dialogangezeigt	Verfügbar in
PR_DISPLAY_NAME	Name, wie im Dialogfeld Adressbuch angezeigt	Outlook, PAB, Schedule+
PR_DISPLAY_NAME_PREFIX	Anrede	Nicht verfügbar in Word
PR_GIVEN_NAME	Vorname	Outlook, PAB, Schedule+
PR_MIDDLE_NAME	Weitere Vornamen	Nicht verfügbar in Word
PR_SURNAME	Nachname	Outlook, PAB, Schedule+
PR_GENERATION	Nachnamenzusatz	Outlook
PR_INITIALS	Initialen	Nicht verfügbar in Word
PR_TITLE	Position	Outlook, PAB, Schedule+
PR_COMPANY_NAME	Firma	Outlook, PAB, Schedule+
PR_OFFICE_TELEPHONE_NUMBER	Geschäftlich (Telefon)	Outlook, PAB, Schedule+
PR_OFFICE2_TELEPHONE_NUMBER	Geschäftlich 2 (Telefon)	Outlook, PAB, Schedule+
PR_BUSINESS_FAX_NUMBER	Fax geschäftlich	Outlook, Schedule+
PR_HOME_TELEPHONE_NUMBER	Privat (Telefon)	Outlook, PAB, Schedule+
PR_HOME2_TELEPHONE_NUMBER	Privat 2 (Telefon)	Nicht verfügbar in Word
PR_HOME_FAX_NUMBER	Fax privat	Outlook
PR_CELLULAR_TELEPHONE_NUMBER	Mobiltelefon	Outlook, PAB, Schedule+
PR_PAGER_TELEPHONE_NUMBER PR_BEEPER_TELEPHONE_NUMBER	Pager	Outlook, PAB, Schedule+
PR_CAR_TELEPHONE_NUMBER	Auto (Telefon)	Outlook
PR_ISDN_NUMBER	ISDN (Telefon)	Outlook
PR_OTHER_TELEPHONE_NUMBER	Weitere (Telefon)	Outlook
PR_PRIMARY_FAX_NUMBER	Weiteres Fax	Outlook, PAB
PR_DEPARTMENT_NAME	Abteilung	Outlook, PAB, Schedule+
PR_OFFICE_LOCATION	Büro	Outlook, PAB, Schedule+
PR_MANAGER_NAME	Vorgesetzte(r)	Nicht verfügbar in Word
PR_ASSISTANT	Assistent	Outlook
PR_PROFESSION	Beruf	Nicht verfügbar in Word
PR_NICKNAME	Spitzname	Nicht verfügbar in Word
PR_SPOUSE_NAME	Name des Partners	Nicht verfügbar in Word
PR_COMMENT	Kommentarfeld	PAB, Schedule+
PR_EMAIL_ADDRESS	E-Mail	Outlook, PAB, Schedule+
PR_ADDRTYPE	Typ der E-Mail-Adresse	Outlook, PAB, Schedule+
PR_STREET_ADDRESS	Straße	Outlook, PAB, Schedule+
PR_POSTAL_CODE	PLZ	Outlook, Schedule+
PR_LOCALITY	Ort	Outlook, PAB, Schedule+
PR_STATE_OR_PROVINCE	Region	Outlook, PAB, Schedule+
PR_COUNTRY	Land	Outlook, PAB, Schedule+

Beispiel für eine Adressdefinition

```
<PR_GIVEN_NAME> <PR_SURNAME>
<PR_COMPANY_NAME>
<PR_STREET_ADDRESS>
<PR_POSTAL_CODE> <PR_LOCALITY>, <PR_STATE_OR_PROVINCE>
```

! Entgegen einiger deutscher Skripts zu diesem Thema (leider auch von Microsoft selbst), **bringt das Nachformatieren der Adressdefinition nichts**, Zwar können Sie in der Adressdefinition z.B. die Felder Postleitzahl und Ort fett formatieren, Word kümmert dies aber nicht. Die substituierten Felder werden dadurch also **nicht** fett formatiert. Es gibt leider keine implizite Lösung um dies zu erreichen, konkret, Sie müssen die Adresse allenfalls manuell nachformatieren.

AutoText-Eintrag aufnehmen

Um den AutoText-Eintrag aufzunehmen, gehen Sie bitte wie folgt vor:

1. Markieren Sie die ganze Felddefinition.
2. Wählen Sie vom Word-Hauptmenü den Dialog ***Einfügen - AutoText - Neu...***.
3. Als Name für den *AutoText* wergeben Sie in **Word 97 Adreßlayout** und in **Word2000 AdressLayout**.
4. Schliessen Sie den Dialog mit ***Ok*** ab.
5. Falls Sie beim Verlassen von Word gefragt werden, ob die Standardvorlage **Normal.dot** gespeichert werden soll, bejahen Sie das bitte.

Werkzeug für Word2000 Benutzer zur Festlegung der Adressdefinition

Benutzer und Benutzerinnen von **Word2000** finden ein Werkzeug von Microsoft, welches das Definieren des AutoText-Eintrages *AdressLayout* etwas vereinfacht. Sie finden in der Dokumentenvorlage **Macros9.dot** ein entsprechendes dialoggeführtes Makro. Falls sich die genannte Vorlage nicht auf Ihre Office2000-CD befindet (was wahrscheinlich ist), können Sie die Vorlage vom deutschen Server von Microsoft herunter laden. Der Link für diesen Download ist unten eingeblendet.

Nun kommt es aber etwas dick. Die lieben Leute, welche dieses Makro auf deutschsprachige Bedürfnisse angepasst haben, haben einen entscheidenden Fehler gemacht. Zwar haben Sie das Drumherum vom Englischen ins Deutsche übersetzt, das wichtigste haben sie jedoch vergessen. Sie erstellen nämlich ein AutoText-Eintrag mit dem Namen *AddressLayout*. Diese englische Schreibweise verhindert aber, dass das Makro in der deutschen Office Version funktioniert. Genauer gesagt, und sogar noch tückischer: das Makro funktioniert zwar, allerdings kommt diese Adressdefinition anschliessend nie zum Tragen. Wie wir inzwischen wissen, muss dieser Eintrag in **Word2000** **AdressLayout** lauten.

Bevor Sie das Werkzeug von Microsoft also brauchen können, müssen Sie es anpassen. Zum Glück beschränkt sich diese Anpassung auf eine einzige Code-Zeile. Alternativ könnten Sie natürlich auch nachdem Sie das Makro ausgeführt haben, den Namen des AutoText-Eintrages über den Word-Dialog ***Extras - Vorlagen und Add-Ins... - Organisieren...*** von AddressLayout nach AdressLayout umbenennen. Diese Lösung ist für mich aber zweite Wahl, da Sie jedes Mal daran denken müssen, wenn Sie dieses Makro einsetzen. Es ist natürlich möglich, dass Microsoft diesen Lapsus früher oder später beseitigen wird, Sie sollten diese Sache aber überprüfen.

Makro einrichten

Um das besprochene Makro bei Ihnen einzurichten, gehen Sie wie folgt vor:

1. Laden Sie sich die Dokumentenvorlage mit dem Namen **Macros9.dot** vom Microsoft Server herunter. Die Adresse für den Download ist unten eingeblendet.
2. Führen Sie einen Doppelklick auf die Download-Datei aus, damit wird die Vorlage auf Ihren Rechner kopiert.
3. **Öffnen** Sie nun die Vorlage.
4. Vom Word-Hauptmenü wählen Sie den Dialog Extras - Vorlagen und Add-Ins... - Organisieren....
5. Wechseln Sie im erscheinenden Dialog in den Reiter ***Makroprojektelemente***.
6. **Kopieren** Sie die beiden Komponenten **frmModifyAddressLayout** und ModifyAddressLayout in die **Normal.dot**.
7. Beenden Sie den Dialog mit der Schaltfläche ***Schliessen***.
8. Wechseln Sie in die VBA-Umgebung in dem Sie die Tastenkombination **[Alt + F11]** betätigen.
9. Lassen Sie sich den Projekt-Explorer anzeigen in dem Sie die Tastenkombination **[Strg + R]** betätigen.
10. Erweitern Sie in diesem Explorer zuerst die Anzeige von **Normal** und anschliessend von **Formulare**.
11. Führen Sie auf den Eintrag **frmModifyAddressLayout** einen Rechtsklick aus und wählen Sie im Kontextmenü den Eintrag **Code anzeigen**.
12. Suchen Sie nach der Code-Zeile: *ALAutoText = "AddressLayout"* und korrigieren Sie diese. Es sollte heissen: **ALAutoText = "AdressLayout"**.
13. Kehren Sie zum Wordfenster zurück und schliessen Sie Word.
14. Falls Sie aufgefordert werden, die Dokumentenvorlage Normal.dot zu speichern, kommen Sie dieser Aufforderung nach.

Makro ausführen

Um das Makro auszuführen und so eine neue Adressdefinition einzurichten, gehen Sie bitte wie folgt vor:

1. Wählen Sie vom Word-Hauptmenü den Dialog ***Extras - Makro - Makros...***.
2. Wählen Sie in der Liste das Makro mit dem Namen **ModifyAddressLayoutStart**.
3. Betätigen Sie die Schaltfläche ***Ausführen***.
4. Der restliche Vorgang ist dann dialoggeführt und eigentlich selbsterklärend.

Weiterreichende Bedürfnisse (Office Automation)

Sollte das hier beschriebene Ihre Bedürfnisse nicht oder nur teilweise abdecken, dann ist wohl Programmierung angesagt. Falls Sie eine neuere Version von MS Outlook einsetzten, finden Sie im unten aufgeführten Link Beispiele wie Sie unter Einsatz von VBA auf den Kontakte-Ordner von MS Outlook zugreifen können. Die Beispiele bedienen sich einer Mimik, welche sich *Office Automation* nennt. Damit ist es möglich, sowohl den Kontakte-Ordner, wie auch die Objektbibliotheke von MS Outlook in Ihrem Code, welcher nicht in MS Outlook ausgeführt wird, verfügbar zu machen.

Eigene Links

► **Wie kann ich auf die Adressen im Outlook Kontakte-Ordner zugreifen?**
► **Von MS Word auf MS Outlook2000 mit Office Automation zugreifen**
► **Wieso kann ich nicht auf das Adressbuch von Outlook Express 4 oder 5 zugreifen.**

Links zu diesem Thema

Microsoft

Deutsch

► **Microsoft Word2000-Zusatzmakros: Macros9.dot**

Englisch

► **WD97: How to Modify the Layout of an Address Book Entry**
► **WD97: Some Contact List Fields Not Available to Word**
► **OL97: Fields Missing Inserting a Contact into Word**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/AL.htm

René Probst, Oktober 1999

Änderungen

- 2. März 2001 - Link zu Office Automation

Wie kann ich alle Kopf- und Fusszeilen eines Dokumentes löschen

◄ Zurück zum Hauptindex

Problemstellung

> 1. Ich möchte alle Kopf- und Fusszeilen in einem bestehenden Dokument loswerden. Wie stelle ich dies an?
>
> 2. Ich möchte die Seitenzahlen in allen Kopf- und Fusszeilen in einem bestehenden Dokument loswerden. Wie stelle ich dies an?

Vorbemerkung

- Ein Dokument kann potentiell sehr viele Kopf- und Fusszeilenbereiche besitzen. Die gesamte Zahl berechnet sich wie folgt: *Anzahl Abschnitte x 2 x 3*.
- Da ein Dokument eine unbestimmte Anzahl von **Abschnitten** aufweisen kann und die Kopf- und Fusszeilenbereiche der einzelnen Abschnitte entkoppelt werden können, ist das Total nach oben offen. Maximal sind es sechs pro Abschnitt. Der Multiplikator zwei steht deshalb, weil ein Dokument sowohl einen Kopf- sowie einen Fusszeilenbereich aufweisen kann. Der Multiplikator drei steht deshalb, weil jeder Abschnitt im äussersten Falle drei Kopf- und Fusszeilenbereiche umfassen kann, namentlich: **Erste Seite**, **Ungerade Seiten** und **Gerade Seiten**.
- Möchten Sie alle Kopf- und Fusszeilen in einem Dokument löschen, bietet sich eine Makrolösung an. Dieses Makro könnte wie folgt aussehen:

VBA-Code

```
Sub AlleKopfundFusszeilenLöschen()
  Dim Abschnitt As Section
  For Each Abschnitt In ActiveDocument.Sections
    For i = 1 To 3
      Abschnitt.Headers(i).Range.Delete
      Abschnitt.Footers(i).Range.Delete
    Next i
  Next
End Sub
```

(Löscht alle Kopf- und Fusszeilen im Dokument.)

Möchten Sie Kopf-/und Fusszeilen zwar erhalten, aber alle Seitenzahlen in diesen Abschnitten in einem Durchgang löschen, sieht die Makrolösung wie folgt aus:

```
Sub AlleSeitenzahlenInKopfUndFusszeileLöschen()
  Dim Abschnitt As Section
  Dim Feld as Field
  For Each Abschnitt In ActiveDocument.Sections
    For i = 1 To 3
      For Each Feld In Abschnitt.Headers(i).Range.Fields
        If Feld.Type = 26 Or Feld.Type = 33 Then Feld.Delete
      Next
      For Each Feld In Abschnitt.Footers(i).Range.Fields
        If Feld.Type = 26 Or Feld.Type = 33 Then Feld.Delete
      Next
    Next i
  Next
End Sub
```

(Löscht Seitenzahlen in allen Kopf- und Fusszeilen des Dokumentes.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/KFLoesch.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich Kopf- und Fusszeilen um 90 Grad drehen?

◄ Zurück zum Hauptindex

Problemstellung

> Ich habe eine Excel-Tabelle im Querformat in ein Word-Dokument eingefügt. Damit die Tabelle genügend Platz zum atmen hat, habe ich eine Seite im Querformat in mein Word-Dokument eingefügt. Dies habe ich dadurch erreicht, dass ich einen Abschnittswechsel - Nächste Seite in das Dokument eingefügt habe.
>
> Nun stehe ich aber vor einem weiterem Problem: die Ausrichtung meiner Kopf- und Fusszeile stimmt in diesem Abschnitt nicht. Wie kann ich den Kopf- und Fusszeilenbereich um 90 Grad drehen?

Vorbemerkung

Microsoft beharrt auf der Feststellung, dass Kopf- und Fusszeilenbereich immer die selbe Ausrichtung wie der übrige Text haben soll. Nun gut, wahrscheinlich ist alles andere etwas zu kompliziert für die Entwickler von Word. Wie auch immer, Sie müssen sich hier selbst helfen. Sowohl das Strukturelement **Textfeld** wie auch das Gestaltungsmittel **Tabelle** ermöglich es Ihnen, einen Text um 90 Grad zu drehen.

Lösung

Mittels Textfeld

1. Wechseln Sie in die Seitenlayoutansicht.
2. Wählen Sie vom Word-Hauptmenü den Dialog ***Einfügen - Textfeld***. Damit ändert sich das Aussehen des Zeigers.
3. Zeichnen Sie am rechten Blattrand ein Textfeld in geeigneter Grösse.
4. Markieren Sie das Textfeld am Rand und führen Sie dann einen Rechtsklick auf das Textfeld aus.
5. Im Kontextmenü wählen Sie nun **Textfeld formatieren**.
6. Wechseln Sie in das Register **Farben und Linien.**.
7. Im Kombinationsfeld unter *Linie - Farbe* wählen Sie **Keine Linie**.
8. Klicken Sie nun auf das Textfeld und wählen Sie in der kontextbezogenen Symbolleiste die Schaltfläche **Textrichtung ändern**.
9. Kopieren Sie nun den gewünschten Text in das Textfeld oder schreiben Sie den gewünschten Text in das Textfeld.

Mittels Tabelle

1. Wechseln Sie in die Seitenlayoutansicht.
2. Wählen Sie vom Word-Hauptmenü den Dialog ***Tabelle - Tabelle zeichnen***. Damit ändert sich das Aussehen des Zeigers
3. Zeichnen Sie am rechten Blattrand eine einzellige Tabelle in geeigneter Grösse.
4. Wählen Sie vom Word-Hauptmenü den Dialog ***Tabelle - Markieren - Tabelle***.
5. Wählen Sie wiederum vom Word-Hauptmenü den Dialog ***Format - Rahmen und Schattierungen....***.
6. Wechseln Sie in das Register **Rahmen.**.
7. Wählen Sie bei *Einstellungen:* **Ohne**.
8. Führen sie einen Rechtsklick auf die Tabelle aus und wählen Sie im Kontextmenü den Befehl **Textausrichtung**....
9. Im nun erscheinenden Dialog sorgen Sie dafür, dass der Text um 90 Grad gedreht wird.
10. Kopieren Sie nun den gewünschten Text in die Zelle oder schreiben Sie den gewünschten Text in die Zelle.

Links zu diesem Thema

Microsoft

► **WD97: How to Add a Portrait Page Number to a Landscape Page**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/KFDrehen.htm

René Probst, Februar 2000

Änderungen

- --- nil ---

Wie kann ich ein Word-Dokument spiegelverkehrt ausrucken?
Wie kann ich Text in einem Word-Dokument auf den Kopf stellen?

◄ Zurück zum Hauptindex

Problem

> 1. Ich muss ein Dokument spiegelverkehrt ausdrucken. Wie erreiche ich dies?
>
> 2. Ich möchte Tischkärtchen erstellen, welche dann in der Mitte gefaltet werden sollen und sich wie eine Toblerone-Verpackung präsentieren sollten. Dafür muss ich den Text zwei mal haben, das zweite Mal allerdings auf dem Kopf stehend.

Antwort zum spiegelverkehrtem Druck

MS Word bietet dazu leider keine Option.

Alternativen zum spiegelverkehrtem Druck

- Drucken Sie den Text auf Klarsichtfolien und legen Sie den Ausdruck verkehrt herum in einen Photokopierer.
- Gewisse Drucker-(Treiber) bieten eine entsprechende Option an.
- Mit Hilfe des kleinen Programms **MS Paint**, welches in Windows integriert ist, ist es möglich Text zu spiegeln und dann den gespiegelten Text als Grafik wieder in das Word-Dokument zu übernehmen. Das ist kein grosses Kunststück, wenn mir MS Word auch beim ersten Versuch mit einer Schutzverletzung in der *winword.exe* abgeschmiert ist und ich nun halt von Neuem diesen Text schreiben darf (*It's Microsoft!*). Gehen Sie wie folgt vor:
    1. Kopieren Sie den fraglichen Teil in die Zwischenablage.
    2. Starten sie über Start - Programme - Zubehör das Programm mit dem Namen **MS Paint**.
    3. Wählen Sie im Hauptmenü von MS Paint ***Bearbeiten - Einfügen*** .
    4. Sie werden nun allenfalls gefragt, ob das Bild vergrössert werden soll, damit der Inhalt in der Zwischenablage Platz auf dem Bild findet. Bejahen Sie diese Frage,
    5. Wählen Sie nun immer noch in MS Paint den Befehl ***Bild - Drehen/Spiegeln*** und wählen Sie im nachfolgenden Dialog die Option **Horizontal Spiegeln**.
    6. Kopieren Sie nun den gespiegelten Text in die Zwischenablage und setzen sie deren Inhalt im Word-Dokument ein.

! Dieser Text liegt nun als Grafik vor und ist somit nicht mehr editierbar!

Text auf den Kopf stellen

MS Word bietet dazu leider keine direkte Option an.

Alternative zum kopfständigen Ausdruck

Mittels WordArt

Da WordArt-Objekte frei gedreht werden können, kann man diese Objekte natürlich auch auf den Kopf stellen. Gehen Sie wie folgt vor:

1. Lassen Sie sich die Symbolleiste mit dem Namen **Zeichnen** anzeigen.
2. Betätigen Sie auf dieser Symbolleiste die Schaltfläche mit dem Namen **WordArt einfügen**.
3. Wählen Sie einer der vorgeschlagenen Formate aus und bestätigen Sie Ihre Wahl mit ***Ok***.
4. Im nun folgenden Dialogfenster schreiben Sie den Text Ihrer Wahl und bestätigen Ihre Eingabe mit ***Ok***.
5. Führen Sie nun einen Rechtsklick auf des eingefügte Objekt aus.
6. Wählen Sie im Kontextmenü den Befehl ***WordArt formatieren...***.
7. Wechseln Sie in den Reiter **Position**.
8. Typen Sie im Feld *Drehung* **180** ein und bestätigen Sie diese Eingabe mit ***Ok***.

Dann wäre da noch die letzte Hürde zu meistern, nämlich das Entfernen dieses völlig werbewirksamen 3D-Effektes des WordArt-Objektes, welches etwa so schön aussieht, wie mein erstes Experiment nachdem man mir das perspektivische Zeichnen beigebracht hatte. Gehen Sie dazu etwa wie folgt vor:

1. Markieren Sie das WordArt-Objekt.
2. Lassen Sie sich die Symbolleiste mit dem Namen **WordArt** anzeigen.
3. Klicken Sie auf die Schaltfläche mit dem Namen **WordArt-Form** und wählen Sie dann bei den Formen den Eintag **Nur Text**.
4. Führen Sie einen Doppelklick auf das WordArt-Objekt aus und passen Sie allenfalls Schriftart, Schriftgrösse. Fett- und Kursivformatierung an.
5. Lassen Sie sich die Symbolleiste mit dem Namen **Zeichnen** anzeigen.
6. Klicken Sie auf die Schaltfläche mit dem Namen **3D** und wählen Sie dann **Kein 3D**.
7. Klicken Sie nun auf die Schaltfläche mit dem Namen **Schatten** und wählen Sie dann **Keinen Schatten**.
8. Führen Sie einen Rechtsklick auf das WordArt-Objekt aus und wählen Sie im Kontextmenü ***WordArt formatieren...***.
9. Wechseln Sie nun in den Reiter **Farben und Linien**.
10. Stellen Sie unter ***Ausfüllen - Farbe*** die Farbe **Schwarz** ein.

Damit sollte nun Ihr Text ziemlich normal aussehen, allerdings wie gewünscht auf dem Kopf stehen.:

Mittels Textfeldern mit gegenseitiger Textausrichtung

Etwas zielgerichteter ist diese Aufgabenstellung zu bewältigen, wenn Sie mit zwei Textfeldern arbeiten, deren Textrichtung um je 90 Grad gedreht ist. Drehen Sie eines der Textfelder im Uhrzeigersinn und das andere gegen den Uhrzeigersinn, damit sich der Effekt nicht aufhebt sondern zu 180 Grad addiert.

Hier eine Schritt- für Schritt-Anweisung dazu.

1. Erstellen Sie ein neues Word-Dokument, am Besten im Querformat.
2. Fügen Sie zwei Textfelder mit der selben Grösse in das Dokument ein und positionieren Sie die Textfelder auf der selben Höhe. Zwischen den Textfeldern soll nur ein sehr kleiner Abstand bestehen.
3. Setzen Sie die Einfügemarke in das linke Textfeld und wählen Sie vom Word-Hauptmenü den Dialog ***Format - Textausrichtung***.
4. Wählen Sie die Textausrichtung, welche von **Oben nach Unten** zeigt.
5. Setzen Sie die Einfügemarke in das rechte Textfeld und wählen Sie abermals den Dialog ***Format - Textausrichtung***.
6. Wählen Sie für dieses Textfeld die Textausrichtung, welche von **Unten nach Oben** zeigt.
7. Setzen Sie die Einfügemarke wieder in das linke Textfeld und wählen Sie vom Word-Hauptmenü den Dialog ***Format - Absatz - Einzüge und Abstände***.
8. Stellen Sie im Kombinationsfeld mit dem Namen *Ausrichtung* den Wert **Zentriert** ein.
9. Wiederholen Sie den letzten Schritt für das Textfeld auf der rechten Seite.

Damit der Schriftzug nicht zu arg am First des werdenden Daches klebt, sollten Sie den Textabstand bezogen auf das Textfeld noch etwas verändern. Gehen Sie wie folgt vor::

1. Markieren Sie den Rand des linken Textfeldes und führen Sie ein Rechtsklick aus.
2. Wählen Sie im Kontextmenü den Befehl ***Textfeld formatieren*** und wechseln Sie in den Reiter **Textfeld**:
3. Vergrössern Sie den Wert im Kombinationsfeld **Innerer Seitenrand - Rechts:**.
4. Wiederholen Sie diesen Vorgang für das rechte Textfeld, vergrössern Sie aber dort den **Innerer Seitenrand - Links:**

Um die beiden Textfelder zu gruppieren und die Rahmen unauffällig zu machen, können Sie wie folgt vorgehen:

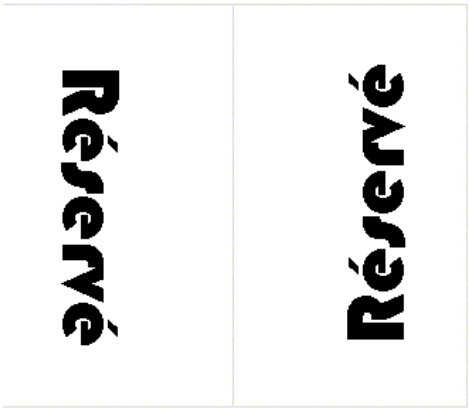
1. Markieren Sie den Rand des linken Textfeldes, halten Sie die **[Umschalt-Taste]** gedrückt und markieren Sie nun auch das rechte Textfeld..
2. Führen Sie einen Rechtklick auf den Rahmen einer der beiden Textfelder aus und wählen Sie im Kontextmenü den Befehl ***Gruppierung - Gruppierung***.
3. Wählen Sie nun wieder im Kontextmenü den Befehl **Objekt formatieren...** und wechseln Sie in den Reiter **Farben und Linien**.
4. Stellen Sie im Kombinationsfeld *Linie - Farbe:* die Farbe **Grau 25%** ein.

VBA-Routine

Nachfolgend eine Routine, welches Ihnen die Erstellung und Formatierung solcher gespiegelten Textfelder abnimmt. Sie werden zuerst aufgefordert, Länge und Breite des Tischkärtchens anzugeben. Bei der Höhe ist die Höhe des gefalteten Tischkärtchens gemeint. Der VBA-Code erstellt dann zuerst ein neues Dokument und platziert das Kärtchen oben links im Dokument. Anschliessend können Sie Text und Aussehen des Textes noch Ihren Bedürfnissen entsprechend anpassen.

Falls Sie vor der Ausführung des Makros Text im Dokument markieren, dann wird dieser Text eingesetzt und auch die Formatierung dieses markierten Textes übernommen. Ist im Dokument nichts markiert, dann kommen die Einstellungen im Kopf des Prozesses zum tragen.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl




```
Private Const Titel = "Tischkärtchen"
Private Const sText = "Ihr Text"
Private Const sBreite = 60
Private Const sHöhe = 35
Private Const sSchriftart = "Arial"
Private Const sFett = True
Private Const sKursiv = False
Private Const sUnterstrichen = False

Sub Tischkärtchen()
  Flag = 0
  tBreite = sBreite
  tHöhe = sHöhe
  While Flag = 0
    Breite = InputBox("Geben Sie die Breite des Tischkärtchens in Milimeter ein.", _
       Titel, tBreite)
    If Breite = "" Then End
    If IsNumeric(Breite) Then
      If Breite <= 160 And Breite >= 16 Then
        Flag = 1
        While Flag = 1
         Höhe = InputBox("Geben Sie die Höhe des Tischkärtchens in Milimeter ein.", _
             Titel, tHöhe)
          If Höhe = "" Then End
          If IsNumeric(Höhe) Then
            If Höhe <= 120 And Höhe >= 12 Then
              Flag = 2
            Else
              MsgBox "Die Höhe muss zwischen 12 und 120 mm liegen", _
                 vbInformation, Titel
              tHöhe = Höhe
            End If
          Else
           MsgBox "Die Höhe muss zwischen 12 und 120 mm liegen", vbInformation, Titel
            tHöhe = Höhe
          End If
        Wend
      Else
        MsgBox "Die Breite muss zwischen 16 und 160 mm liegen", vbInformation, Titel
        tBreite = Breite
      End If
      Else
        MsgBox "Die Breite muss zwischen 16 und 160 mm liegen", vbInformation, Titel
        tBreite = Breite
    End If
  Wend
  Dim nDoc As Document, oRange As Range
  Dim Form(1) As Shape
  Set oRange = Selection.Range
  Wahl = Selection.Type
```

```
  Set nDoc = Documents.Add
  With nDoc.Sections(1)
    .PageSetup.Orientation = wdOrientLandscape
     x = .PageSetup.LeftMargin
     y = .PageSetup.TopMargin
     h = MillimetersToPoints(Breite)
     b = MillimetersToPoints(Höhe)
  End With
  Set Form(0) = nDoc.Shapes.AddTextbox(3, x, y, b, h)
  Set Form(1) = nDoc.Shapes.AddTextbox(2, x + b, y, b, h)
  Form(0).TextFrame.MarginRight = b * 2 / 5
  Form(1).TextFrame.MarginLeft = b * 2 / 5
  If Wahl = 2 Then
    tText = oRange.Text
    tSchriftart = oRange.Font.Name
    tFett = oRange.Font.Bold
    tKursiv = oRange.Font.Italic
    tUnterstrichen = oRange.Font.Underline
    tGrad = oRange.Font.Size
  Else
    tText = sText
    tSchriftart = sSchriftart
    tFett = sFett
    tKursiv = sKursiv
    tUnterstrichen = sUnterstrichen
    tGrad = Int(b / 3)
  End If
  For i = 0 To 1
    With Form(i).TextFrame.TextRange
      .InsertAfter tText
      .ParagraphFormat.Alignment = wdAlignParagraphCenter
      .Font.Name = tSchriftart
      .Font.Bold = tFett
      .Font.Italic = tItalic
      .Font.Underline = tUnterstrichen
      .Font.Size = tGrad
    End With
  Next i
  Set Form(0) = _
     ActiveDocument.Shapes.Range(Array(Form(0).Name, Form(1).Name)).Group
  Form(0).Line.ForeColor.RGB = RGB(222, 222, 222)
  Form(0).Line.Weight = 0.25
End Sub
```

(Diese Prozedur erstellt ein Tischkärtchen mit Word-Mitteln)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Spiegel.htm

René Probst, Oktober 1999

Änderungen

- 13. Juni 2000 - Methode MS Paint
- 13. Juni 2000 - Gespiegelte Textfelder

Word bietet mir nur 16 Schriftfarben an. Kann ich das ändern?

◄ Zurück zum Hauptindex

Problem

> **Im Dialog *Format - Zeichen... - Schriftfarbe:* werden mir nur 16 Farben angeboten. Diese Farben sind mir aber zu grell. Gibt es hier keine anderen Farben bzw. Zwischentöne?**

Antwort

☹ Was **Word 97** und seine Vorgänger betrifft, sind Sie auf die 16 Farben, welche Ihnen MS Word zur Verfügung stellt beschränkt. **Word2000** dagegen bietet Ihnen mehr als **16 Millionen Farben bzw. Schattierungen** zur Auswahl an.

Anmerkungen

- Im Gegensatz zu **Word 97** bietet **Word2000** auch die Möglichkeit Wörter andersfarbig zu unterstreichen.
- Auch für Hintergrundsfarben gehört die Beschränkung der 16 Farben der Vergangenheit an. Auch hier gilt: Sie können aus dem Vollen schöpfen.
- In der neusten Ausgabe von Word können Sie Farben also nach dem bewährten Muster **RGB** (Rot - Grün - Blau) mixen. Jede Grundfarbe kann dabei einen Wert von **0** bis **255** einnehmen. Je höher die Werte, desto heller die Farbe. RGB(0,0,0) ergibt schwarz, RGB(255,255,255) ergibt weiss, RGB(127,127,127) ergibt ein mittleres grau. Die sechs Grundfarben aus dem Farbkreis erreichen Sie wie folgt:
  - ☺ Rot - RGB(255,0,0)
  - ☺ Orange - RGB(255,127,0)
  - ☺ Gelb - RGB(255,255,0)
  - ☺ Grün - RGB(0,255,0)
  - ☺ Blau - RGB(0,0,255)
  - ☺ Violett - RGB(127,127,0)

☛ Eine typische Pastellfarbe erreichen Sie dadurch, dass Sie zuerst die gewünschte Grundfarbe wählen und anschliessend in gleichen Teilen noch die anderen beiden Farben in abgeschwächter Intensität dazu geben.

VBA

Diese Änderung bezüglich der Farbskala muss sich auch in VBA ausdrücken. Die Eigenschaft **ColorIndex** und ihre Konstanten (**wdColorIndex**) sind unter **Word2000** immer noch gültig beschränken sich aber weiterhin auf die 16 Farben. Neu hinzugekommen ist die Eigenschaft **Color**. Die Syntax sieht dabei wie folgt aus: `Selection.Font.Color = RGB(87, 255, 127).`

Achtung: Diese Art der Farbgebung ist nicht rückwärtskompatibel, wird also von früheren Versionen von Word nicht unterstützt, provoziert aber einen Syntaxfehler.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Farben.htm
René Probst, Oktober 1999

Änderungen
- 29. Dezember 1999 - RGB-Methode

Wie bringe ich eine Faltmarke in ein Word-Dokument ein?

◄ Zurück zum Hauptindex

Problemstellung

> Damit mir das genaue Falten eines Briefes etwas leichter fällt, möchte ich an geeigneter Stelle im Dokument eine Falzmarke einbringen. Leider habe ich keine entsprechende word-integrierte Funktion gefunden.

Lösung

Sie können eines der vier folgenden Makros benutzen. Das erste und dritte Makro setzt eine Faltmarke an den linken Rand des Dokumentes, das zweite und vierte Makro setzt sowohl eine Faltmarke, wie auch ein Hilfsstrich, genau in der Hälfte des Dokuments, welcher eine präzise Lochung zulässt.

Implementierung und Aufruf

Um diese Lösung allgemein verfügbar zu machen, gehen Sie bitte wie folgt vor:

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot**.
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
3. Wählen Sie dort vom Hauptmenü den Befehl ***Einfügen - Modul.***
4. Kopieren Sie eines der Makros oder gleich mehrere in das leere Fenster.
5. Speichern Sie die Standardvorlage und verlassen Sie Word.
6. Starten Sie Word nochmals.
7. Wählen Sie vom Word-Hauptmenü den Befehl ***Extras - Makro - .Makros...***.
8. In der erscheinenden Liste der Makros, markieren Sie **FalzmarkeEinfügen** bzw. **FalzmarkeEinfügenNurErsteSeite**.
9. Betätigen Sie nun die Schaltfläche ***Ausführen***.
10. **Drucken** Sie zum Testen eine oder mehrere Seiten.

Anmerkungen

- Das Makro verankert die Hilfsstriche bewusst in der Kopfzeile. Damit lassen sie sich nur sehr schlecht unabsichtlich verschieben.
- Dieses Skript schlägt vier Lösungen vor. Das erste Makro fügt eine Faltmarke auf allen Seiten eines vielleicht mehrseitigen Dokumentes ein, das zweite Makro fügt sowohl eine Faltmarke, sowie auch eine Hilfslinie für die Zweifachlochung auf **jeder Seite** des Dokumentes ein. Das dritte Makro fügt eine Faltmarke, allerdings **nur auf der ersten Seite** des Dokumentes ein. Das vierte und letzte Makro fügt sowohl eine Faltmarke wie auch eine Hilfslinie für die Zweifachlochung ausschliesslich auf der ersten Seite des gedruckten Dokumentes ein.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

Faltmarke auf jeder Seite des gedruckten Dokumentes

```
Sub FaltmarkeEinfügen()
  'Die Länge des Striches beträgt 0,9 - 0,5 = 0,4 cm
  Dim oKz As HeaderFooter, FM As Shape
  Set oKz = ActiveDocument.Sections(1).Headers(1)
  Set FM = oKz.Shapes.AddLine(CentimetersToPoints(0.5), _
                              CentimetersToPoints(10.4), _
                              CentimetersToPoints(0.9), _
                              CentimetersToPoints(10.4))

  FM.Line.Weight = 0.25
  FM.LockAnchor = True
End Sub
```

(Fügt eine Faltmarke in das Dokument ein. Diese Hilfsstriche erscheinen auf *jeder Seite* des Dokumentes.)

Falt- und Lochmarke auf jeder Seite des gedruckten Dokumentes

```
Sub FalzUndLochmarkeEinfügen()
  Dim oKz As HeaderFooter, FM As Shape, LM As Shape
  Set oKz = ActiveDocument.Sections(1).Headers(1)
  'Die Länge des Striches (Faltmarke) beträgt 1,2 - 0,5 = 0,7 cm
  Set FM = oKz.Shapes.AddLine(CentimetersToPoints(0.5), _
                              CentimetersToPoints(10.4), _
                              CentimetersToPoints(1.2), _
                              CentimetersToPoints(10.4))

  FM.Line.Weight = 0.25
  FM.LockAnchor = True
  'Die Länge des Striches (Lochmarke) beträgt 0,9 - 0,5 = 0,4 cm
  Set LM = oKz.Shapes.AddLine(CentimetersToPoints(0.5), _
                              CentimetersToPoints(14.85), _
                              CentimetersToPoints(0.9), _
                              CentimetersToPoints(14.85))
  LM.Line.Weight = 0.25
  LM.LockAnchor = True
End Sub
```

(Fügt eine Faltmarke und eine Markierung für die Zweier-Lochung in das Dokument ein. Diese Hilfsstriche erscheinen auf *jeder Seite* des Dokumentes.)

Faltmarke nur auf der ersten Seite des gedruckten Dokumentes

```
Sub FalzmarkeEinfügenNurErsteSeite()
  'Die Länge des Striches beträgt 0,9 - 0,5 = 0,4 cm
  Dim oKz As HeaderFooter, FM As Shape
  Rand = ActiveDocument.Sections(1).PageSetup.LeftMargin
  ActiveDocument.Sections(1).PageSetup.DifferentFirstPageHeaderFooter = True
  Set oKz = ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage)
  Set FM = oKz.Shapes.AddLine(CentimetersToPoints(0.5) - Rand, _
                              CentimetersToPoints(10.4), _
                              CentimetersToPoints(0.9) - Rand, _
                              CentimetersToPoints(10.4), oKz.Range)

  FM.Line.Weight = 0.25
  FM.LockAnchor = True
End Sub
```

(Fügt eine Faltmarke in das Dokument ein. Dieser Hilfsstriche erscheint ausschliesslich auf *der ersten Seite* des Dokumentes.)

Falt- und Lochmarke nur auf der ersten Seite des gedruckten Dokumentes

```
Sub FalzUndLochmarkeEinfügen()
  Dim oKz As HeaderFooter, FM As Shape, LM As Shape
  Rand = ActiveDocument.Sections(1).PageSetup.LeftMargin
  ActiveDocument.Sections(1).PageSetup.DifferentFirstPageHeaderFooter = True
  Set oKz = ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage)
  'Die Länge des Striches (Faltmarke) beträgt 1,2 - 0,5 = 0,7 cm
  Set FM = oKz.Shapes.AddLine(CentimetersToPoints(0.5) - Rand, _
                              CentimetersToPoints(10.4), _
                              CentimetersToPoints(1.2) - Rand, _
                              CentimetersToPoints(10.4))

  FM.Line.Weight = 0.25
  FM.LockAnchor = True
  'Die Länge des Striches (Lochmarke) beträgt 0,9 - 0,5 = 0,4 cm
  Set LM = oKz.Shapes.AddLine(CentimetersToPoints(0.5) - Rand, _
                              CentimetersToPoints(14.85), _
                              CentimetersToPoints(0.9) - Rand, _
                              CentimetersToPoints(14.85))
  LM.Line.Weight = 0.25
  LM.LockAnchor = True
End Sub
```

(Fügt sowohl eine Faltmarke wie auch einen Hilfsstrich für die Zweier-Lochung in das Dokument ein. Diese Hilfsstriche erscheinen auf j*eder Seite* des Dokumentes.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Falz.htm

René Probst, Oktober 1999

Änderungen

- 17. Juni 2000 - Problematik Erste Seite anders
- 16. November 2000 - Linien statt Textfelder, da diese in WD2000 standardmässig anders ausschauen, als in WD97 (dickere Linie)
- 16. November 2000 - Problematik mit "Erste Seite anders" Dank *Ibby* (Ibrahim Elnazak) gelöst

Wie kann ich kann ich den Dateiname eines Dokumentes mit Pfadangabe in die Kopf- oder Fusszeile setzen?

◄ Zurück zum Hauptindex

Problemstellung

> Ich möchte den Dateinamen des aktiven Dokumentes mit Pfadangabe in der Kopf- oder Fusszeile referenzieren. Wird die Datei umbenannt, soll sich diese Angabe neu synchronisieren.

Lösung

Sie erreichen dies auf einfache Weise mittels einer Feldfunktion oder noch schneller über den Dialog ***Einfügen - AutoText***.

Einfügen über AutoText-Dialog

Word stellt Ihnen diesen AutoText-Eintrag bereits zur Verfügung. Dieser ist in der Standardvorlage *Normal.dot* enthalten. Um ihn zu nutzen, gehen Sie wie folgt vor:

1. Wechseln Sie in die **Kopf- oder Fusszeilenansicht**.
2. Wählen Sie vom Word-Hauptmenü den Dialog ***Einfügen - AutoText***.
3. Wählen Sie in der sich nun präsentierenden Liste den Eintrag **Dateiname und Pfad**.

Anmerkungen

- Falls sich der Zeiger nicht in der Kopf- oder Fusszeile befindet, also die Formatvorlage **Standard** aktiv ist, wird noch ein Untermenü dazwischen geschaltet. In diesem Falle müssen Sie folgendes wählen: ***Einfügen - AutoText - Kopf-/ Fusszeile - Datei mit Pfad***.
- Statt *Datei und Pfad* können Sie selbstverständlich auch nur **Datei** wählen, falls die Sie keine Pfadangabe möchten.
- Damit die Pfadangabe angezeigt werden kann, muss das Dokument mindestens einmal gespeichert worden sein, andernfalls kann Word den Pfad natürlich nicht ermitteln.

Einfügen über Feld-Dialog

1. Wechseln Sie in die **Kopf- oder Fusszeilenansicht**.
2. Wählen Sie vom Word-Hauptmenü den Dialog ***Einfügen - Feld...***.
3. Markieren Sie in der linken Liste *Kategorien:* den Eintrag **Dokumentinformationen**.
4. Markieren Sie in der rechten Liste *Feldnamen:* den Eintrag **Dateinamen**.
5. Betätigen Sie die Schaltfläche ***Optionen...***.
6. Wechseln Sie im nun folgenden Dialog in den Reiter ***Spezifische Schalter***.
7. Markieren Sie in der Liste den Schalter **/p** (es gibt keinen anderen).
8. **Wichtig:** Betätigen Sie die Schaltfläche ***Hinzufügen***.
9. Schliessen Sie den Dialog mit ***Ok*** ab. (2x)

Anmerkungen

- Wenn Sie nur den Dateinamen nicht aber den Pfad einfügen möchten, dann fällt das Setzen des Schalters selbstverständlich weg.
- Damit die Pfadangabe angezeigt werden kann, muss das Dokument mindestens einmal gespeichert worden sein, andernfalls kann Word den Pfad natürlich nicht ermitteln.
- Beachten Sie, dass diese Feldfunktion in **Word 97** **Dateiname** und in **Word2000** **FileName** heisst.
- Die Möglichkeit den Dateiname ohne Dateierweiterung (.doc) einzufügen, ist leider **nicht** gegeben.

Feldfunktion in der Dokumentvorlage einbringen

Wenn Sie die hier beschriebene Feldfunktion in der Dokumentvorlage eingeben, auf welche Ihre Schriftstücke basieren, dann wird die Funktion beim Erstellen eines neuen Dokumentes auf dieses vererbt.

Um dies zu erreichen, öffnen Sie die gewünschte Dokumentvorlage bringen die Feldfunktion wie hier beschrieben dort ein und speichern die Vorlage. Wiederholen Sie diesen Vorgang allenfalls für andere Dokumentvorlagen, welche Sie einsetzen.

VBA-Code

Folgender VBA-Code druckt zwei Exemplare des aktiven Dokumentes. Das erste Exemplar wird ohne Fusszeile gedruckt (zum **Versand** and den Kunden), das zweite Exemplar enthält in der Fusszeile den Dokumentenname einschliesslich Pfad (für die **Ablage**).

Der Code geht davon aus, dass keine weitere Informationen in der Fusszeile enthalten sind. Nach dem Druck, wird das Feld, welche den Pfad anzeigt wieder gelöscht.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub MitUndOhneDateinamen()
  Dim oDoc As Document
  Dim oRange As Range
  Set oDoc = ActiveDocument
  oDoc.PrintOut
  Set oRange = oDoc.Sections(1).Footers(1).Range
  oRange.Fields.Add Range:=oRange, Type:=wdFieldFileName, Text:="\p"
  oDoc.PrintOut Background:=False
  oRange.Delete
End Sub
```

(Zwei Exemplare werden gedruckt; ein Exemplar ohne Fusszeile zum Versand und ein Duplikat, welches den Pfad des Dokumentes in der Fusszeile enthält - zur Ablage.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/DatName.htm

René Probst, Oktober 1999

Änderungen

- 21. Dezember 1999 - Makro Dateiname im Duplikat
- 22. Januar 2001 - Feldfunktion in der Dokumentvorlage eingeben

Wie kann ich ein Symbol kreieren (z.B. ein x mit Überstrich, wie es in statistischen Berechnungen gebraucht wird)?

◄ Zurück zum Hauptindex

Problemstellung

> Ich brauche für mein Dokument einige Sonderzeichen, welche in keiner Schrift enthalten sind und welche ich auch nicht über den Dialog *Einfügen - Sonderzeichen...* in mein Dokument einbringen kann. Konkret brauche ich ein kleingeschriebenes *x mit Überstrich* und ein kleingeschriebenes *b mit einem Dach* (accent circonflexe). Mit dem Formeleditor habe ich eher schlechte Erfahrungen gemacht. Ich möchte dieses Unterprogramm für die nur sporadischen Fälle, in der ich solche Zeichen benötige, nicht gebrauchen.

Lösung

Sie können dafür eine Feldfunktion mit dem spezifischen Schalter **Overlay** (englisch für überlagern) bzw. dessen Abkürzung **O** verwenden und sich damit selber die fehlenden Zeichen basteln. Der Trick bei der Sache ist, dass durch geeignete Steuerung im Dokument zwei individuelle Zeichen übereinandergelegt werden und in dieser Form angezeigt bzw. gedruckt werden. Um diese Zeichen anschliessend besser zugänglich zu machen, verwenden Sie das **AutoKorrektur**-Feature von Word.

Wie für die meisten Feldfunktionen, gibt es zwei Wege diese einzubringen. Der erste Weg führt über den Word-Dialog *Einfügen - Feld...*. Alternativ haben Sie die Möglichkeit, die ganze Feldfunktion direkt einzugeben.

Feldcode mittels Dialog einbringen

Um ein x mit Überstrich zu basteln, gehen Sie wie folgt vor:

1. Wählen sie vom Word-Hauptmenü den Befehl ***Einfügen - Feld...***.
2. In der linken Liste des erscheinenden Dialoges markieren Sie die *Kategorie* **Formel und Ausdrücke**.
3. In der rechten Liste wählen Sie den *Feldnamen* **Formel** **Word 97** bzw. **Eq** **Word2000**.
4. Betätigen Sie nun die Schaltfläche ***Optionen...***.
5. Wählen Sie in der Liste den *Schalter* **\O()** aus.
6. **Wichtig:** betätigen Sie die Schaltfläche ***Hinzufügen***.
7. Verlassen Sie diesen Unterdialog mit ***Ok***.
8. Platzieren Sie nun den Zeiger im Textfeld zwischen die beiden **runden Klammern**.
9. Schreiben Sie zwischen die Klammern die Zeichenfolge **`x;`**.
10. Schalten Sie die Taste **NumLock** ein und betätigen Sie die Tastenfolge **[Alt + 0175]**. Benutzen Sie dabei zur Eingabe der Zahlen den Ziffernblock rechts auf Ihrer Tastatur.
11. Der gesamte Ausdruck sieht nun wie folgt aus: **`FORMEL \O(x;¯)`** **Word 97** bzw. **`EQ\O(x;¯)`** **Word2000**.
12. Schliessen Sie den Dialog mit ***Ok*** ab.
13. Das Resultat ist ein x mit Überstrich. Falls Ihnen Word stattdessen den Feldcode anzeigt, sollten Sie die Tastenkombination **[Alt + F9]** betätigen.

Feldcode direkt einbringen

Nun nochmals das selbe, wobei der Feldcode aber direkt eingegeben wird

1. Betätigen Sie die Tastenkombination **[Alt + F9]** um in die Feldcodeansicht zu wechseln.
2. Generieren Sie das Gerüst für den Feldcode, indem Sie die Tastenkombination **[Strg + F9]** betätigen.
3. Sie sollten nun ein **geschweiftes Klammerpaar** sehen, die Einfügemarke steht mitten drin.
4. Schreiben Sie nun die Zeichenfolge: **`FORMEL \O (x;`**.
5. Schalten Sie die Taste **NumLock** ein und betätigen Sie die Tastenfolge **[Alt + 0175]**. Benutzen Sie zur Eingabe der Zahlen den Ziffernblock rechts auf Ihrer Tastatur.
6. Schliessen Sie den Ausdruck mit einer runden Klammer ab. Der ganze Ausdruck sollte nun wie folgt aussehen: **`{ FORMEL \O (x;¯) }`** Word 97 bzw. **`{ EQ \O (x;¯) }`** **Word2000**.
7. Betätigen Sie die Tastenkombination **[Alt + F9]** um in die Feldcodeansicht auszuschalten. Das Resultat ist ein x mit Überstrich.

Anmerkungen zur Bauanleitung für bei einem b mit Dach

- Der ANSI-Code für den *accent circonflexe* lautet **0136**.
- Damit unser Werk bei dem hohen Buchstaben "b" gut aussieht, sollten wir das Dach noch etwas höher ansetzen. Dies Erreichen wir wie folgt:
    1. In der **Feldcodeansicht** markieren wir das Dach.
    2. Vom Word-Hauptmenü wählen wir den Dialog ***Format - Zeichen ...- Zeichenabstand***.
    3. Beim Kombinationsfeld **Position** wählen wir den Wert **Höherstellen** um **5 pt.** . Variieren Sie diesen Wert nach Ihrem Gutdünken, falls Sie besonders klein oder gross schreiben.

Anmerkungen

- ! Vergessen Sie nicht, dass Sie nach jeder Korrektur des Feldcodes das Feld aktualisieren müssen, damit der neue Versuch zum Tragen kommt. Sie erreichen dies, wenn sie das Feld markieren und die Taste **[F9]** betätigen.
- Beachten Sie, dass die Feldfunktion in **Word 97** **FORMEL** heisst, in **Word2000** aber **EQ**.

Vollbrachtes Werk als AutoKorrektur-Eintrag verewigen

Wenn Sie diese Aktion etwa 333 Mal durchgeführt haben, beherrschen Sie diesen Vorgang aus dem Handgelenk. Zwar langsam aber stetig wächst bei Ihnen der Wunsch es sich für die Zukunft etwas bequemer einzurichten. Dazu biete sich die AutoKorrektur von Word an. Gehen Sie wie folgt vor:

1. Markieren Sie das Feld.
2. Vom Word-Hauptmenü wählen Sie den Dialog ***Extras - AutoKorrektur...***.
3. Vergeben sie im Feld **Ersetzen:** ein Kürzel, welches Ihnen als Gedächtnisstütze dient.
4. Schliessen Sie den Dialog mit ***Ok*** ab.

Vergessen Sie nicht, wenn Sie beim Beenden von Word gefragt werden, ob die Standardvorlage "Normal.dot" gespeichert werden soll, dieses zu bejahen. Von nun an können Sie diese Feldfunktion unter dem vergebenen Kürzel ansprechen. Sowie Sie anschliessend die Leertaste betätigen oder einen Zeilenumbruch einfügen, wird das Kürzel vollautomatisch mit der Feldfunktion ersetzt.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Symbolk.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich Nullwerte beim Berechnung unterdrücken?

◄ Zurück zum Hauptindex

Problemstellung

> In einer nicht vollständig ausgefüllten Tabelle führe ich Berechnungen durch, wie das im unten aufgeführten Beispiel demonstriert wird. In den nicht ausgefüllten Zeilen werden mir bei der Berechnung des Totalbetrages hässliche Nullen angezeigt. Wie kann ich diese Nullen unterdrücken?

Beispiel

Anzahl	Artikel	Stückpreis	Preis
3	Schreibblöcke	3,95	11,85
4	Kugelschreiber	2,50	10,00
	Bleistifte	0,85	0
	Radiergummi	2,25	0

Lösung

Sie können das gewünschte Zahlenformat gleich an die Formel anhängen. Verwenden Sie statt der Formel: { `= Produkt(a2;c2)` } einfach Die Formel:

```
{ = Produkt(a2;c2) \# "0,00;-0,00;''"}
```

Anmerkungen

- Der Text für den Formatschalter besteht aus drei Teilen. Der erste Teil kommt bei positiven Beträgen zum tragen, der zweite bei negativen Werte. Der dritte Teil, welcher aus zwei Hochkommas (Quotes) besteht, entfernt die Null. Das Ganze Format muss dann noch in Gänsefüsschen eingeschlossen werden.
- Die geschweiften Klammern dürfen nicht konventionell eingegeben werden sondern müssen mit der Tastenkombination **[Strg + F9]** generiert werden.
- Der Ausdruck zur Berechnung eines Produktes heisst in **Word2000** nicht *Produkt* sondern **Product**. Das neuste Kind von Microsoft kennt hier also nur noch englische Ausdrücke.
- Wenn Ihr Windows auf schweizerische Verhältnisse abgestimmt ist lautet das Dezimalzeichen **.** (Punkt) und nicht **,** (Komma), wie im hier aufgeführten Beispiel.
- **Wichtig:** Vergessen Sie nicht nach jeder Korrektur des Feldcodes das Feld zu aktualisieren. Nur so erreichen Sie, dass der Code neu interpretiert wird. Um das Feld zu aktualisieren, markieren Sie es und betätigen dann die Taste **[F9]**.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/NullWert.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich auf fünf (Rappen) genau runden?

◄ Zurück zum Hauptindex

Problemstellung

> Ich habe eine Tabelle in der ich Berechnungen durchführe. In der Schweiz werden Geldbeträge ausschliesslich auf fünf Rappen gerundet dargestellt. Aus einem Betrag von 23,12 soll also 23,10 werden und bei 23,13 soll 23,15 berechnet werden. Wie muss die Formel (der Feldcode) aussehen, welche diese spezielle Art von Rundung ausführt.

Lösung

Allgemein

Um den Geldbetrag von z. B. 24.13 umzuwandeln verwenden Sie folgenden Feldcode:

```
{ = Int (24,13 * 20 + 0,5) / 20 \# "0,00;-0,00"}
```

Tabelle

Um den Geldbetrag in der Zelle **a1** umzuwandeln verwenden Sie folgenden Feldcode:

```
{ = Int ( { = a1 } * 20 + 0,5) / 20 \# "0,00;-0,00"}
```

Textmarke

Um den Geldbetrag in der Textmarke **Text1** bzw. in einem **Formularfeld** umzuwandeln verwenden Sie folgenden Feldcode:

```
{ = Int ( Text1 * 20 + 0,5) / 20 \# "0,00;-0,00"}
```

Anmerkungen

- Der Text für den Formatschalter besteht aus zwei Teilen. Der erste Teil kommt bei positiven Beträgen zum tragen, der zweite bei negativen Werten. Das Ganze Format muss dann noch in Gänsefüsschen stehen.
- Die geschweiften Klammern dürfen nicht konventionell eingegeben werden, sondern müssen mit der Tastenkombination **[Strg + F9]** generiert werden.
- Wenn Ihr Windows auf schweizerische Verhältnisse abgestimmt ist, lautet das Dezimalzeichen **.** (Punkt) und nicht **,** (Komma).

Links zu diesem Thema

Microsoft

► Runden auf ein Vielfaches z. B. 0,05 (Rappen)

Eigene Links

► Wie kann ich Nullwerte beim Berechnung unterdrücken?

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Fuenfer.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Datums- und Stundenberechnung mit Feldern, Textmarken, Formularfeldern und Tabellen

◄ Zurück zum Hauptindex

Problemstellung

1. Ich möchte ein Haltbarkeitsdatum als dynamisch berechnendes Feld in einer Dokumentenvorlage haben. Wie sieht die Syntax des Feldcodes dazu aus?

2. Eine Feldfunktion Kalenderwoche kennt Word nicht. Wie kann ich selber eine solche Feldfunktion implementieren?

3. Ich möchte mit Textmarkenfeldern (Datums)berechnungen durchführen. Wie stelle ich das an?

4. Ich möchte mit konventionelle Formulartextfeldern (Datums)berechnungen durchführen. Wie stelle ich das an?

5. Ich möchte meine Präsenszeiten in einer Word-Tabelle führen und berechnen. Wie muss ich vorgehen?

Diese Seite

▼ Vorbemerkungen
▼ Einfache Datumsberechnung mittels dynamischem Feldcode
▼ Komplexer Feldcode zur Datumsberechnung von Chris Woodman
▼ Allgemeine Felder (Platzhalter)
▼ Kalenderwoche Feld-Funktion selbst gemacht
▼ Mit Textmarken rechnen
▼ Mit konventionellen Formulartextfeldern rechnen
▼ Datumsberechnungen in einer Word-Tabelle
▼ Links zu diesem Thema

Vorbemerkungen

▲ Zurück zum Anfang

Word-Felder beherrschen die Datumsarithmetik nicht. Mit Biegen und Brechen ist es aber dennoch möglich, einfache Datumsberechnungen durchzuführen. Mit monströsen Felddefinitionen ist es auch möglich, immerhin bis zu 28 Tagen nach vorne zu rechnen, wie die Arbeit von **Chris Woodman** beweist. Chris stellt auch gleich einen komfortablen Dialog zur Verfügung, welcher es erlaubt, solche Monsterfelddefinitionen vollautomatisch zu erstellen. Dieses Skript wird auch darauf noch zurückkommen.

Auch mit Textmarkenfeldern und konventionellen Formularfeldern gelingen mit vernünftigen Aufwand nur sehr einfache Berechnungen. Sobald zwei unterschiedliche Monate bei der Berechnung involviert sind, hört es mit der Herrlichkeit bereits auf. Die Berechnung liefert dann nämlich unsinnige Ergebnisse. Für komplexere Berechnungen in diesem Bereich, sollten Sie deshalb auf VBA zurückgreifen, welches Ihnen auch in diesem Bereich den Komfort und die konsistente Syntax liefert, wie man dies von Programmiersprachen erwartet.

☛ Falls das Dokument keine weiteren datumsspezifischen Felder enthält, können Sie erwägen, das Systemdatum unter Windows für die Zeit des Ausdruckes zu verändern.

Einfache Datumsberechnung mittels dynamischem Feldcode

▲ Zurück zum Anfang

Folgender Feldcode berechnet das Datum **Heute in zwei Jahren**. Die Schaltjahrproblematik bleibt dabei unberücksichtigt.

Word 97

```
{ ZEIT \@ "t.M." } { = { Zeit \@ "jjjj" } + 2 }
```

Word2000

```
{ TIME \@ "d.M." } { = { TIME \@ "yyyy" } + 2 }
```

Folgender Feldcode berechnet das Datum **des letzen Monats**. Mit dieser Funktion könnten Sie einen sich dynamisch anpassenden Brief im folgenden Stil schreiben: *Heute ist der 1. Februar 1999. Ich erlaube mir, Ihnen die Abrechnung für den Monat Januar 1999 zukommen zu lassen.*

Word 97

```
{ ZEIT \@ "t" }. { = { Zeit \@ "M" } - 1 }. { ZEIT \@ "jjjj" }
```

Word2000

```
{ TIME \@ "d" }. { = { TIME \@ "M" } - 1 }. { TIME \@ "yyyy" }
```

Komplexer Feldcode zur Datumsberechnung von *Chris Woodman*

▲ Zurück zum Anfang

Funktionalität

Wie schon erwähnt, gibt es eine Person, welche genug Können und Geduld hatte, den Feldcode für eine Funktion heraus zu tüfteln, welcher es erlaubt **bis zu 28 Tagen** zu einem vorgegeben Datum hinzu zu rechnen.

Mehr noch, diese Person hat auch eine komfortable Oberfläche entwickelt, welche diesen Feldcode, automatisch generiert. Neben der Verzögerung in Tagen, können Sie auch das Format, in dem das mittels Feldfunktion errechnete Datum erscheinen soll, vorgeben.

Damit Sie sich ein Bild davon machen können, was Chris da geleistet hat, der Feldcode für eine Verzögerung von 14 Tagen (ausgehend von einem beliebigen Datum) sieht dann in **Word2000** wie folgt aus:

```
{ QUOTE "{ SET Delay "14" }{ SET "DaysInMonth" { IF
{ CREATEDATE \@ "MM" } <> 2 { = round(30.575*{ CREATEDATE \@ "MM" }
;0)-ROUND(30.575*{ = { CREATEDATE \@ "MM" } - 1 };0) }{ IF { = MOD(
{ CREATEDATE \@"yy" }; 4 ) } > 0 "28" "29" } } }{ SET "NextMonth" { IF
{ CREATEDATE \@ "MM" } = 12 "1/97" "{ = { CREATEDATE \@ "MM" } + 1 }/97 } }
{ IF { = { REF Delay } + { CREATEDATE \@ "dd" } } <= { DaysInMonth }
{ CREATEDATE \@ "{ = { REF Delay } + { CREATEDATE \@ "dd" } } - MMMM - yyyy" }
{ QUOTE { = { REF Delay } + { CREATEDATE \@ "dd" } - { DaysInMonth } } -
{ NextMonth \@ "MMMM" } - { IF { CREATEDATE \@ "MM" } <> 12 { CREATEDATE \@ "yyyy" }
{ CREATEDATE \@ "{ = 1 + { CREATEDATE \@ "yyyy" } \# "xxxx" }" } } } }" }
```

(Ausgehend von einem beliebigen Datum rechnet folgender Monsterfeldcode z.B. 14 Tage nach vorne. Entwickelt hat diesen Code *Chris Woodman*. Er stellt auch eine Oberfläche zur Verfügung, um den Tatzelwurm zu generieren.)

Bezugsquelle
Am Ende dieses Skriptes ist der Link eingeblendet über welchen Sie die Arbeit von Chris Woodman beziehen können.

Lösung verfügbar machen
Das ist denkbar einfach. Gehen Sie wie folgt vor:

1. Laden Sie die kleine ZIP-Datei herunter (den Link finden Sie am Schluss dieses Skriptes).
2. Entzippen Sie die herunter geladenen Datei. Sie enthält eine einzige Datei mit dem Namen **DelayDat.dot**.
3. Führen Sie einen Doppelklick auf diese Dokumentenvorlage aus.
4. Starten Sie über die Symbolleiste, welche im Dokument enthalten ist, den Dialog.
5. Bestimmen Sie wie viele **Tage** dazu gerechnet werden sollen und geben Sie das **Format** vor, indem das Datum angezeigt werden soll
6. Wenn Sie nun die Schaltfläche ***Ok*** betätigen, wird die Feldfunktion im Dokument eingefügt.

☛ Falls Sie oft solche Feldcodes erstellen müssen, kopieren Sie die Dokumentvorlage, welche Chris bereit stellt in das Office **StartUp-Verzeichnis**.

Datumsberechnung mittels VBA-Code
Wesentlich flexibler und konsistenter kriegen Sie diese Sache in den Griff, wenn Sie VBA-Code einsetzen. Optional können Sie den VBA-Code nachdem Sie ihn getestet habe, in einem Prozess mit dem reservierten Name **AutoOpen** bzw. **AutoNew** einsetzen. Damit steht einer automatisierten Lösung eigentlich nichts im Wege.

Sie finden gleich anschliessend zahlreiche Beispiele und Anregungen, wie dies verwirklicht werden kann.

Allgemeine Felder (Platzhalter)
▲ Zurück zum Anfang

Einleitung
In diesem Kapitel geht es darum, wie Sie das Ergebnis einer Berechnung, welche mit VBA erreicht wurde, an geeigneter Stelle in das Dokument einbringen. Am einfachsten wäre dies mit `Selection.TypeText`-*Methode* zu erreichen. Diese Methode setzt dann das Ergebnis dort ein, wo sich die Einfügemarke befindet.

Wenn das Dokument (bzw. die Dokumentvorlage) eine Art Maske darstellen soll, ist diese Methode allerdings nicht der Weisheit letzter Schluss. Viel mehr sind hier Plathalter gefragt, welche Sie zur Entwurfszeit in die Dokumentvorlage einbringen können. Diese Platzhalter sollen dann später, z.B. wenn ein neues Dokument auf Basis dieser Vorlage erstellt wird, mit dem Ergebnis der VBA-Funktion versorgt werden.

Dazu eigenen sich Textmarkenfelder, welche im nächsten Kapitel besprochen werden aber auch Dokumentvariablen oder Dokumenteigenschaften. Der Umgang mit Textmarkenfelder ist übrigens wesentlich einfacher zu erreichen, als die Methode welche nun besprochen wird.

Dieses Kapitel beschränkt sich vorerst auf Dokumentvariablen. Der Grund, wieso ich Dokumenteigenschaften ebenfalls ins Spiel bringe, ist in einem Fehlverhalten von Word 97 zu suchen. Word 97 schmiert Ihnen nämlich ab, sollten Sie versuchen, einen Platzhalter statt in den Hauptteil des Dokumentes in die Kopf-/Fusszeile oder ein Textfeld einzufügen. Erst mit **Word2000** ist dieses Problem von Microsoft gelöst worden.

! Sollten Sie Word 97 benutzen und beabsichtigen Platzhalter ebenfalls in den genannten Dokumentteilen einzusetzen, dann lesen Sie auch das *Kapitel: Lösung mit Dokumenteigenschaften*. Der Weg, welcher dahin führt, ist sehr ähnlich, wie der Weg, welcher jetzt aufgezeigt wird. Erwaten Sie deshalb nicht, dass dort nochmals alles so ausführlich wie jetzt erklärt wird.

Der Weg zu einer benutzerdefinierten Feldfunktion

Sie können in Ihrem Dokument oder Ihrer Dokumentvorlage eine beliebige Anzahl von Variablen definieren. Tun Sie dies in der Dokumentvorlage, werden diese Variablen an das Dokument, welche auf Basis dieser Dokumentvorlage erstellt wird vererbt. Im weiteren brauchen Sie einen Plathalter im Dokument oder in der Dokumentvorlage und schliesslich benötigen Sie noch ein Stück VBA-Code, welcher einerseits die Berechnung durchführt und andererseits diese Dokumentvariablen versorgt. Ich versuche diesen etwas komplexe aber sehr flexibel einsetzbare Mimik zusammenzufassen:

1. Sie fügen der Dokumentvorlage eine Dokumentvariable hinzu. Dabei handelt es sich vorerst einzig um eine Definition (eine Art Felddefinition). Diese Variable wird beim Vorgang *Datei - Neu* an das neu generierte Dokument vererbt. Dieser Vorgang geschieht in der Regel genau einmal und ist nur unter Einsatz von VBA möglich. Wir sind nun so weit, dass die Dokumentvorlage (und später ihre Kinder), von dieser Variablen Kenntnis hat. Sehen tun Sie noch nichts, es handelt sich lediglich um eine Veranlagung.
2. Nun fügen Sie der Dokumentvorlage an geeigneter Stelle einen Plathalter (man kann auch sagen eine Referenz auf diese Dokumentvariable) hinzu. Auch dieser Platzhalter wird an das künftige Dokument vererbt. Auch dieser Vorgang geschieht in der Regel genau einmal und zwar indem Sie der Dokumentvorlage ein bestimmtes Feld an geeigneter Stelle hinzufügen. Soweit haben wir nun die Veranlagung geschaffen und bestimmt, an welcher Stelle im Dokument unsere Früchte erscheinen sollen. Nun müssen wir noch liefern.
3. Nun wird eine VBA-Routine (Makro), welche unsere Berechnung durchführt angestossen und unser Plathalter mit dem Ergebnis versorgt. Dieser Anstoss kann automatisiert geschehen, z.B. zum Zeitpunkt wenn ein neues Dokument generiert wird, oder es kann manuell passieren, in dem der Benutzer unser Makro aufruft. Im Gegensatz zu den vorhergehenden Vorgängen ist dies ein sich ständig wiederholender Vorgang, also eine dynamischer Sache. Dies ist auch das Ziel, denn wenn ich zwei Tage vom aktuellen Datum abziehe, dann erhalte ich morgen schon ein anders Ergebnis als heute.

Mit diesen Schritten, haben wir genau das erreicht, was eine von Word intern zur Verfügung gestellte Feldfunktion auch tut. Der Unterschied liegt also beim Lösungsweg. Bei einer Word-integrierten Feldfunktion fallen die Schritte eins und drei weg.

Szenario

Im folgenden wird nun ein vorgegebenes Szenario durchgespielt. Berits das erste von uns erarbeitete und einfache Beispiel wird für die meisten Fälle in der Praxis genügen. Das Beispiel wird dann noch dahingehen verfeinert, dass auch Spezialfälle abgedeckt sind.

Aufgabenstellung ist es, eine benutzerdefinierte dynamische Feldfunktion zu erstellen, welche die aktuelle Kalenderwoche anzeigen kann. Dabei können wir, was den funktionelle Teil betrifft, auf eine sehr kurze und bewährte VBA-Funktion zurückgreifen.

Dokumentvariable in die Dokumentvorlage einbringen

Wie Eingangs erwähnt wurde, ist dies ausschliesslich mittels VBA möglich. Dabei handelt es sich aber um keine grosse Sache. Zuerst einigen wir uns auf den Namen dieser Variable und nennen Sie **Kalenderwoche**.

Als nächstes wird unsere Dokumentvorlage über ***Datei - Öffnen*** geöffnet. Am besten Sie achten darauf, dass im weiteren keine Word-Dokumente geöffnet sind, sonst landet der nun folgende Code gar noch in der falschen Datei. Wechseln sie nun in den VBA-Editor, in dem Sie die Tastenkombination **[Alt+F11]** betätigen. Wählen Sie dann ***Einfügen - Modul*** und kopieren Sie den folgenden Code in das leere Fenster, welches sich geöffnet hat:

```
Sub DokVarHinzfügen()
  ActiveDocument.Variables.Add Name:="Kalenderwoche"
End Sub
```

(Die Variable mit dem Namen "Kalenderwoche" wird der Dokumentvorlage bekannt gemacht.)

Lassen Sie dieses Makro nun ausführen. Diese Aktion hinterlässt keine sichtbaren Spuren. Wenn Sie überprüfen wollen, ob es geklappt hat, rufen Sie das Makro gleich nochmals auf, dann erfolgt nämlich eine Fehlermeldung, welche besagt, dass das Ding bereits vorhanden ist. Sie erkenne daran, dass dieser Aufruf nur einmal im Leben der Vorlage geschehen muss bzw. geschehen soll.

Platzhalter in der Dokumentvorlage einfügen

Die nächste Aktion, dürfte Ihnen geläufig sein. Sie sollen eine Feldfunktion an geeigneter Stelle (also an der Stelle im Hauptdokument, wo die Kalenderwoche erscheinen soll) in die Dokumentvorlage einbringen. Gehen Sie im einzelnen wie folgt vor:

1. Setzen Sie in der Dokumentvorlage die Einfügemarke an den Ort, an der die Laufnummer erscheinen soll.
2. Vom Word-Hauptmenü wählen Sie nun den Dialog ***Einfügen - Feld...***.
3. In der linken Liste mit der Bezeichnung *Kategorien:* markieren sie den Eintrag **Dokumentautomation**.
4. In der rechten Liste mit der Bezeichnung *Feldnamen:* markieren Sie den Eintrag **DokVariable** falls Sie **Word 97** benutzen oder aber **DocVariable** falls Sie **Word2000** benutzen.
5. Als nächstes sollten Sie das Textfeld in diesem Dialog ergänzen. Setzen Sie also die Einfügemarke hinter den Feldnamen DokVariable (bzw. DocVariable), achten Sie darauf, dass eine Leerstelle vorhanden ist und tippen sie **"Kalenderwoche"** und zwar **inklusive Gänsefüsschen**. Der ganze Ausdruck heisst nun also: **DokVariable "Kalenderwoche"** bzw. **DocVariable "Kalenderwoche"**.
6. Schliessen Sie nun den Dialog mit ***Ok*** ab und sichern Sie die Dokumentvorlage.

Kalenderwochen-Funktion einbringen und die Dokumentvariable mit dem Ergebnis versorgen

Im folgenden nun der gesamte Code, welcher benötigt wird, die aktuelle Kalenderwoche aus dem aktuellen Datum zu berechnen. Zusätzlich wird dann unsere Dokumentvariable mit dem Ergebnis versorgt. Da nach jeder Änderung eines Feldwertes, das **Feld aktualisiert** werden muss (normalerweise erreichen Sie dies mit der Taste [F9] - wir streben aber eine Automation an), kommt noch eine Code-Zeile hinzu.

Da wir möchten, dass diese Funktion automatisch aufgerufen wird, sobald ein neues Dokument auf Basis unserer Dokumentvorlage generiert wird, muss der Name des Prozesses **AutoNew** lauten.

Kopieren Sie also den folgenden Code ebenfalls in die fragliche Dokumentvorlage. Ob Sie dabei ein neues Modul eröffnen, oder diesen Code hinter den Dreizeiler kopieren, welcher sich bereits in der Vorlage befindet, ist eigentlich gleichgültig.

```
Sub AutoNew()
  ActiveDocument.Variables("Kalenderwoche").Value = KWoche(Date)
  ActiveDocument.Fields.Update
End Sub

Private Function KWoche(d As Date) As Integer
'Kalenderwochen nach DIN 1355
'Algorythmus von Christoph Kremer, Aachen
  Dim t&
  t = DateSerial(Year(d + (8 - Weekday(d)) Mod 7 - 3), 1, 1)
  KWoche = (d - t - 3 + (Weekday(t) + 1) Mod 7) \ 7 + 1
End Function
```

(Die Funktion *KWoche* wird immer dann aufgerufen, wenn ein neues Dokument auf Basis der Dokumentvorlage generiert wird. Die Dokumentvariable mit dem Namen *Kalenderwoche* wird dann mit dem Ergebnis versorgt und die Felder im Hauptteil des Dokumentes werden aktualisiert.)

Damit hätten wir das nötigste beisammen. Im folgenden wird nun dieser Code verfeinert und wird dann folgende zusätzliche Ausstattungsmerkmale aufweisen:

- Der Code stellt eigenmächtig fest, ob die Dokumentvariable mit dem Namen *Kalenderwoche* bereits definiert ist. Wenn nicht, wird diese Definition erstellt.
- Es wird sichergestellt, dass die Dokumentvariable mit dem Name *Kalenderwoche* auch bei einem späteren Öffnen des Dokumentes aktualisiert wird.
- Der Wert für die Dokumentvariable mit dem Name *Kalenderwoche*, wird auch dann aktualisiert, wenn sich die entsprechende Felddefinition ausserhalb des Hauptteils des Dokumentes befindet, also z.B. in der Kopf-/Fusszeile oder in einem Textfeld.

Kalenderwoche Feld-Funktion selbst gemacht

▲ Zurück zum Anfang

Lösung mit Dokumentvariable

Hier als die verfeinerte Lösung. Kopieren Sie folgenden Code in ein neues Modul der fraglichen Dokumentvorlage und testen Sie dann die Lösung.

Einsatzgebiet

☺ Diese Variante kann und sollte immer dann angewendet werden, wenn Sie entweder mit **Word2000** arbeiten. Grundsätzlich sollten Sie auch diese Variante wählen, wenn Sie mit Word 97 arbeiten, es sei denn, Sie möchten den Platzhalter in der Kopf-/Fusszeile oder in einem Textfeld einsetzen (an Stelle des Hauptteils des Dokumentes).

☹ Diese Variante kann **nicht** angewendet werden, falls Sie mit **Word 97** arbeiten und den Platzhalter in die **Kopf-/Fusszeile oder in ein Textfeld** einfügen wollen. In diesem Falle sollten Sie auf die Lösung mit Dokumenteigenschaften ausweichen, welche gleich anschliessend kurz vorgestellt wird. Word 97 schmiert reproduzierbar ab, falls Sie für diese Dokumententeile Dokumentenvariablen verwenden. (*Great, Microsoft!*)

Anmerkung: Sollten Sie hier direkt eingestiegen sein, dann lesen Sie bitte zuerst das Kapitel:

► Allgemeine Felder (Platzhalter)

```
Sub AutoNew()
  CheckExistance "Kalenderwoche"
  ActiveDocument.Variables("Kalenderwoche").Value = KWoche(Date)
  FieldRefreshLoop
End Sub

Sub AutoOpen()
  CheckExistance "Kalenderwoche"
  ActiveDocument.Variables("Kalenderwoche").Value = KWoche(Date)
  FieldRefreshLoop
End Sub

Private Function KWoche(d As Date) As Integer
'Kalenderwochen nach DIN 1355
'Algorythmus von Christoph Kremer, Aachen
  Dim t&
  t = DateSerial(Year(d + (8 - Weekday(d)) Mod 7 - 3), 1, 1)
  KWoche = (d - t - 3 + (Weekday(t) + 1) Mod 7) \ 7 + 1
End Function

Private Function FieldRefreshLoop()
  Dim Teil As Range, Feld As Field
  For Each Teil In ActiveDocument.StoryRanges
    For Each Feld In Teil.Fields
      If Feld.Type = wdFieldDocVariable Then Feld.Update
    Next
  Next
End Function

Private Function CheckExistance(vName As String)
  Dim v As Variable
  Flag = False
  For Each v In ActiveDocument.Variables
    If v.Name = vName Then
      Flag = True
      Exit For
    End If
  Next
  If Flag = False Then ActiveDocument.Variables.Add Name:=vName
End Function
```

(Die Funktion *KWoche* wird immer dann aufgerufen, wenn ein neues Dokument auf Basis der Dokumentvorlage generiert wird oder dieser Klon geöffnet wird. Die *Dokumentvariable* mit dem Namen *Kalenderwoche* wird dann mit dem Ergebnis versorgt und die Felder in allen Dokumententeile werden aktualisiert.)

Lösung mit Dokumenteigenschaft

Hier also nun die Lösung mit einer Dokumenteigenschaft statt einer Dokumentvariable. Ich führe nochmals den Grund auf, wieso ich auch diese Lösung zur Debatte stelle:

Einsatzgebiet

☹ Die Variante, welche vorher besprochen wurde und mit einer Dokumentvariable arbeitet kann dann nicht verwendet werden, wenn folgende beiden Bedingungen zutreffen: Sie arbeiten Sie mit **Word 97** und wollen den Platzhalter in die **Kopf-/Fusszeile oder in ein Textfeld** einfügen.

Falls Sie diese Aufgabe für eine heterogene Umgebung erarbeiten (Word 97 und Word2000 gemischt), dann wählen Sie ebenfalls diese Lösung, da sie ja allgemeingültiger ist.

Die Vorarbeiten Sind fast identisch. Sie sollten aber diesmal statt einer Dokumentenvariable eine **Dokumenteigenschaft** als Platzhalter benutzen. Diese Feldfunktion finden Sie unter ***Einfügen - Feld - Dokumentinformation - DokEigenschaft*** **Word 97** bzw. ***DocProperty*** **Word2000**.

Als Name für diese Dokumenteigenschaft vergeben wir einmal mehr **Kalenderwoche**. Der VBA-Code in seiner Endfassung soll jetzt wie folgt aussehen und gehört in die Dokumentvorlage.

```
Sub AutoNew()
  CheckExistance "Kalenderwoche"
  ActiveDocument.CustomDocumentProperties("Kalenderwoche").Value = KWoche(Date)
  FieldRefreshLoop
End Sub

Sub AutoOpen()
  CheckExistance "Kalenderwoche"
  ActiveDocument.CustomDocumentProperties("Kalenderwoche").Value = KWoche(Date)
  FieldRefreshLoop
End Sub

Private Function KWoche(d As Date) As Integer
'Kalenderwochen nach DIN 1355
'Algorythmus von Christoph Kremer, Aachen
  Dim t&
  t = DateSerial(Year(d + (8 - Weekday(d)) Mod 7 - 3), 1, 1)
  KWoche = (d - t - 3 + (Weekday(t) + 1) Mod 7) \ 7 + 1
End Function

Private Function FieldRefreshLoop()
  Dim Teil As Range, Feld As Field
  For Each Teil In ActiveDocument.StoryRanges
    For Each Feld In Teil.Fields
      If Feld.Type = wdFieldDocProperty Then Feld.Update
    Next
  Next
End Function

Private Function CheckExistance(vName As String)
  Dim v As DocumentProperty
  Flag = False
  For Each v In ActiveDocument.CustomDocumentProperties
    If v.Name = vName Then
      Flag = True
      Exit For
    End If
  Next
  If Flag = False Then
    ActiveDocument.CustomDocumentProperties.Add Name:=vName, _
       LinkToContent:=False, Type:=msoPropertyTypeNumber, Value:=0
  End If
End Function
```

(Die Funktion *KWoche* wird immer dann aufgerufen, wenn ein neues Dokument auf Basis der Dokumentvorlage generiert wird oder dieser Klon geöffnet wird. Die *Dokumenteigenschaft* mit dem Namen *Kalenderwoche* wird dann mit dem Ergebnis versorgt und die Felder in allen Dokumententeile werden aktualisiert.)

Mit Textmarken rechnen

▲ Zurück zum Anfang

Voraussetzungen

Um das folgende Makro zu testen, sollten Sie die folgenden Voraussetzungen schaffen:

1. Erstellen Sie ein neues Dokument.
2. Schreiben Sie ein gültiges Datum z.B. *11.03.99* an eine beliebige Stelle im Dokument.
3. Markieren Sie das Datum.
4. Vom Word-Hauptmenü wählen Sie den Dialog ***Einfügen - Textmarke...***.
5. Vergeben Sie der Textmarke den Namen **Eingegangen** und klicken Sie ***Hinzufügen***.
6. Schreiben Sie an einer anderen Stelle ein weiteres gültiges Datum z.B. *01.11.99*.
7. Markieren Sie jetzt auch dieses Datum.
8. Wählen Sie wiederum den Dialog ***Einfügen - Textmarke...***.
9. Vergeben Sie der Textmarke den Namen **Erledigt** und klicken Sie ***Hinzufügen***.
10. Schreiben Sie an einer anderen Stelle im Dokument das Word **Dauer**.
11. Markieren Sie das Wort.
12. Wählen Sie wiederum den Dialog ***Einfügen - Textmarke...***.
13. Vergeben Sie der Textmarke den Namen **Dauer** und klicken Sie ***Hinzufügen***.

VBA-Code (statische Variante)

Kopieren Sie folgenden Code in der VBA-Umgebung in ein neues Modul:

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub TMTageBerechnenStatisch()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Ein = oDoc.Bookmarks("Eingegangen").Range.Text
  Aus = oDoc.Bookmarks("Erledigt").Range.Text
  If IsDate(Ein) And IsDate(Aus) Then
    oDoc.Bookmarks("Dauer").Range.Font.Color = wdColorBlack
    oDoc.Bookmarks("Dauer").Range.Text = _
       DateDiff("d", DateValue(Ein), DateValue(Aus))
  Else
    oDoc.Bookmarks("Dauer").Range.Font.Color = wdColorRed
    oDoc.Bookmarks("Dauer").Range.Text = "Fehler!"
  End If
End Sub
```

(Aus zwei Datumsangaben in Textmarken wird die Differenz berechnet. Das Resultat wird einer dritten Textmarke zugewiesen. Dieser Prozess kann nur einmal ablaufen, da die Zieltextmarke überschrieben wird.)

Anmerkungen

- Um die Position von Textmarken zu sehen, sollten Sie die Option **Textmarken** im Word-Dialog ***Extras - Optionen... - Ansicht - Anzeigen*** aktivieren.
- Die hier zur Verwendung kommenden Textmarken, sollten sowohl links wie rechts mit einer eckigen Klammer begrenzt sein. Sehen Sie nur linksbündig die Klammer, dann liegt es daran, dass Sie vor dem Einfügen der Textmarke keinen Text markiert haben.
- Bitte beachten Sie, dass bei dieser Methode die Zieltextmarke mit dem Namen **Dauer** überschrieben wird. An ihrer Stelle steht nach Ablauf des Prozesses die Datumsdifferenz in Tagen. Die Textmarke selbst, wird dadurch aber zerstört. Ein zweites Mal kann dieser Prozess also nicht mehr korrekt ablaufen. Möchten Sie dies vermeiden, halten Sie sich bitte an das nun folgende Code-Beispiel.
- Sie dürfen auch andere gültige Datumsformate benutzen, als das eingangs vorgeschlagene Format *tt.mm.jj*.

VBA-Code (Dynamische Variante)

Das folgende Code-Beispiel geht von den genau gleichen Voraussetzungen aus wie das eben diskutierte und verfolgt auch fast das gleiche Ziel. Diesmal wird aber sicher gestellt, dass die Zieltextmarke erhalten bleibt. Das Makro kann also dynamisch eingesetzt werden und jedes mal aufgerufen werden, falls entweder das Eingangsdatum oder das "Erledigt-Datum" verändert wurde.

```
Sub TMTageBerechnenDynamisch()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Dim Marke As Range
  Ein = oDoc.Bookmarks("Eingegangen").Range.Text
  Aus = oDoc.Bookmarks("Erledigt").Range.Text
  Set Marke = oDoc.Bookmarks("Dauer").Range
  If IsDate(Ein) And IsDate(Aus) Then
    Marke.Text = DateDiff("d", DateValue(Ein), DateValue(Aus))
  Else
    Marke.Text = "Fehler!"
  End If
  oDoc.Bookmarks.Add Name:="Dauer", Range:=Marke
End Sub
```

(Aus zwei Datumsangaben in Textmarken wird die Differenz berechnet. Das Resultat wird einer dritten Textmarke zugewiesen.)

Mit konventionellen Formulartextfeldern rechnen

▲ Zurück zum Anfang

Noch besser lässt es sich mit konventionellen Formulartextfeldern rechnen, wie Sie diese über die Symbolleiste **Formular** in ein Dokument einbringen können.

Voraussetzungen

Um das folgende Makro zu testen, sollten Sie die folgenden Voraussetzungen schaffen:

1. Erstellen Sie eine neue Dokumentenvorlage.
2. Kopieren Sie das unten abgebildete Makro mit dem Namen **FFTageBerechnen** im VBA-Editor in ein neues Modul.
3. Lassen Sie sich die Symbolleiste **Formular** anzeigen.
4. Fügen Sie an beliebiger Stelle ein **Textformularfeld** in das Dokument ein.
5. Lassen Sie sich über die kontextbezogene Symbolleiste die **Formularfeld-Optionen** anzeigen..
6. Stellen sie im Kombinationsfeld **Typ** den Wert **Datum** ein und passen Sie gegebenenfalls auch das **Datumsformat** im gleichen Dialog an.
7. Im Feld **Textmarke** sollten Sie dem Feld einen Namen vergeben. Nennen Sie das Feld **CheckIn**.
8. Im Kombinationsfeld *Makro beim Verlassen* **Word 97** bzw. *Makro ausführen bei beenden* **Word2000** suchen Sie nun in der Liste nach dem Makro mit dem Namen **FFTageBerechnen** und setzen diesen Wert ein.
9. Schliessen Sie den Dialog mit ***Ok*** ab.
10. Setzen Sie nun die Einfügemarke an eine andere Stele im Dokument und fügen Sie dort ein weiteres Textformularfeld ein.
11. Passen Sie dieses Feld nach den gleichen Voraussetzungen an wie das erste, mit dem wichtigen Unterschied, dass dieses Feld **CheckOut** heissen soll.
12. Setzen Sie nun die Einfügemarke an eine andere Stele im Dokument und fügen Sie dort ein weiteres Textformularfeld ein.
13. Setzen Sie auch hier den Feldtyp auf Datum und passen Sie allenfalls das Datumsformat an.
14. Entfernen Sie nun bei diesem Feld das Häkchen bei **Eingabe zulassen**.
15. Nennen Sie dieses Feld **Nächte** und schliessen Sie den Dialog mit ***Ok*** ab.
16. **Schützen** Sie nun das Formular mit der entsprechenden Schaltfläche auf der kontextbezogenen Symbolleiste.
17. Speichern Sie die Vorlage.
18. Erstellen Sie eines neues Dokument auf der Basis dieser Vorlage und testen Sie das Werk.

VBA-Code (Feld-Exit Prozedur)

Kopieren Sie folgenden Code in der VBA-Umgebung in ein neues Modul und zwar in die Vorlage, welche die Formularfelder enthält.

```
Sub FFTageBerechnen()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  CIn = oDoc.FormFields("CheckIn").Result
  COut = oDoc.FormFields("CheckOut").Result
  If IsDate(CIn) And IsDate(COut) Then
    Nächte = DateDiff("d", DateValue(CIn), DateValue(COut))
  End If
End Sub
```

(Aus zwei konventionellen Formularfeldern wird die Datumsdifferenz berechnet und in ein drittes Formularfeld geschrieben. Diese Prozedur ist als Feld-Exit gedacht.)

Bei den nachfolgenden Beispielen wird aus zwei Zeitangaben die Differenz, ausgedrückt in Stunden und Minuten (erstes Beispiel) bzw. ausgedrückt in Stunden, Minuten und Sekunden (zweites Beispiel), berechnet. Diese Beispiele lehnen sich stark an das vorhergehende Beispiel an. Diesmal lauten die Namen der Formularfelder allerdings anders, nämlich: **Anfang**, **Ende** und **Arbeitszeit**.

```
Sub FFSundenMinutenBerechnen()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Ein = oDoc.FormFields("Anfang").Result
  Aus = oDoc.FormFields("Ende").Result
  If IsDate(Ein) And IsDate(Aus) Then
    mm = DateDiff("n", TimeValue(Ein), TimeValue(Aus))
    hmm = Format(TimeSerial(0, mm, 0), "h:mm")
    oDoc.FormFields("Arbeitszeit").Result = hmm
  End If
End Sub
```

(Aus zwei konventionellen Formularfeldern wird die Zeitdifferenz, ausgedrückt in Stunden und Minuten, errechnet und in ein drittes Formularfeld geschrieben. Diese Prozedur ist als Feld-Exit gedacht.)

```
Sub FFSundenMinutenSekundenBerechnen()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Ein = oDoc.FormFields("Anfang").Result
  Aus = oDoc.FormFields("Ende").Result
  If IsDate(Ein) And IsDate(Aus) Then
    ss = DateDiff("s", TimeValue(Ein), TimeValue(Aus))
    hmmss = Format(TimeSerial(0, 0, ss), "h:mm:ss")
    oDoc.FormFields("Arbeitszeit").Result = hmmss
  End If
End Sub
```

(Aus zwei konventionellen Formularfeldern wird die Zeitdifferenz, ausgedrückt in Stunden, Minuten und Sekunden, errechnet und in ein drittes Formularfeld geschrieben. Diese Prozedur ist als Feld-Exit gedacht.)

Datumsberechnungen in einer Word-Tabelle

▲ Zurück zum Anfang

Dank VBA lässt sich auch in Word-Tabellen ganz gut rechnen, wenn vielleicht auch nicht ganz so **Excel**lent. Das nachfolgende Beispiel geht als Szenario von folgender Tabelle aus:

Szenario

Erstellen Sie ein neues Word-Dokument. Fügen Sie eine dreispaltige Tabelle in das Dokument ein und füllen Sie diese etwa wie folgt aus:

Eingang	Erledigt	Dauer
11.11.97	24.12.99	
14. Dezember 1999	23.01.00	
02.05.26	02.05.27	
11.11.11	12.11.11	
Dauer Total		

Fügen Sie nun den folgenden Code in der VBA-Umgebung in ein leeres Modul, erstellen Sie anschliessend eine Verknüpfung vom Makro auf eine Symbolleiste Ihrer Wahl und testen Sie die Datumsberechnung in der Tabelle in dem Sie das Makro ausführen lassen und allenfalls die Werte in der Tabelle verändern oder auch zusätzliche, analoge Zeilen hinzufügen.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub TabelleTageBerechnen()
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Das Dokument verfügt über keine Tabelle."
    End
  End If
  Dim oTable As Table
  Set oTable = ActiveDocument.Tables(1)
  If oTable.Columns.Count < 3 Or oTable.Rows.Count < 3 Then
    MsgBox "Die Tabelle sollte mindestens 3 Zeilen und drei Spalten aufweisen."
    End
  End If
  i = 0
  For Each Zeile In oTable.Rows
    i = i + 1
    If i > 1 And i < oTable.Rows.Count Then
      Ein = Left(Zeile.Cells(1), Len(Zeile.Cells(1)) - 2)
      Aus = Left(Zeile.Cells(2), Len(Zeile.Cells(2)) - 2)
      If Len(Ein) + Len(Aus) > 0 Then
        If IsDate(Ein) And IsDate(Aus) Then
          Zeile.Cells(3) = DateDiff("d", DateValue(Ein), DateValue(Aus))
          Zeile.Cells(3).Range.Font.Color = wdColorBlack
        Else
          Zeile.Cells(3) = "Fehler!"
          Zeile.Cells(3).Range.Font.Color = wdColorRed
        End If
      Else
        Zeile.Cells(3) = ""
        Zeile.Cells(3).Range.Font.Color = wdColorBlack
      End If
    End If
  Next
  TotalTage = 0
  i = 0
  For Each Zelle In oTable.Columns(3).Cells
    i = i + 1
    If i > 1 And i < oTable.Rows.Count Then
      Wert = Left(Zelle, Len(Zelle) - 2)
      If IsNumeric(Wert) Then
        TotalTage = TotalTage + Wert
      End If
    End If
  Next
  oTable.Rows(oTable.Rows.Count).Cells(3).Range.Text = TotalTage
  oTable.Rows(oTable.Rows.Count).Cells(3).Range.Font.Bold = True
End Sub
```

(Datumsberechnung für eine Tabelle.)

Das Ergebnis sollte sich etwa wie folgt präsentieren:

Eingang	Erledigt	Dauer
11.11.97	24.12.99	773
14. Dezember 1999	23.01.00	40
02.05.26	02.05.27	365
11.11.11	12.11.11	1
Dauer Total		1179

Links zu diesem Thema

▲ Zurück zum Anfang

Microsoft

► Feld mit aktuellem Datum + zwei Jahre bzw. 15 Monate

Chris Woodman

► Delayed Date Field Code Generator

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/DatBer.htm

René Probst, Oktober 1999

Änderungen

- 15. November 2000 - Arbeit von Chris Woodman (Delayed Date Field Code Generator)
- 25. November 2000 - Arbeiten mit Dokumentvariablen
- 25. November 2000 - Kalenderwochefunktion

Wie kann ich eine Fussnotenzahl doppelt vergeben?

◄ Zurück zum Hauptindex

Problemstellung

> Ich suche nach einer Möglichkeit, Mehrfachverweise aus meinem Text auf ein und den selben Fussnotentext einzubringen. Wie schaffe ich das?

Lösung

Sie erreichen dies mit einem **Querverweis**. Das erste Mal wird die Fussnote wie gewohnt eingefügt, für weitere Verweise benutzen Sie dann einen Querverweis. Gehen Sie im einzelnen wie folgt vor:

1. Fügen Sie die Fussnote beim ersten Mal wie gewohnt ein.
2. Setzen Sie die Einfügemarke an diejenige Stelle, an der der zweite Verweis erscheinen soll.
3. Wählen Sie vom Word-Hauptmenü den Dialog ***Einfügen - Querverweis...***.
4. Stellen sie das linke Kombinationsfeld **Verweistyp:** auf den Wert ***Fußnote***.
5. Stellen sie das rechte Kombinationsfeld **Verweisen auf:** auf den Wert ***Fußnotennummer (formatiert)***.
6. Betätigen Sie zuerst die Schaltfläche ***Einfügen*** und dann die Schaltfläche ***Schliessen***.

Anmerkungen

- Falls Ihr Word die Option *Fußnotennummer (formatiert)* nicht kennen sollte, wählen Sie an deren Stelle die Option **Fußnotennummer** und formatieren Sie die Zahl mit der Formatvorlage **Fußnotenzeichen** nach.
- Um die Fussnotennummern zu aktualisieren (z. B. nachdem eine gelöscht wurde), markieren Sie das ganze Dokument und betätigen die Taste **[F9]**. Alternative können Sie auch in die Seitenansicht und wieder zurück wechseln.

Links zu diesem Thema

Microsoft

Deutsch

► Mehrere Verweise auf die gleiche Fußnote

Englisch

► WD97: How to Create Multiple References to the Same Footnote

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/FNQuer.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich die Fussnotentrennlinie formatieren bzw. löschen?

◄ Zurück zum Hauptindex

Problem

> Ich arbeite an einem wissenschaftlichen Text. Das Seitenlayout sieht links einen Rand von 3 cm und rechts einen Rand von 5 cm vor. Die Trennlinie vor dem Fussnotenbereich passt nun aber gar nicht zu diesem Layout. Wie kann ich diesen Strich anders formatieren bzw. löschen?

Antwort

Dieser Vorgang ist in der Word-Online-Hilfe beschrieben. Gehen Sie wie folgt vor:

1. Vom Word-Hauptmenü wählen Sie ***Ansicht - Normal***.
2. Nochmals über das Menü **Ansicht** wechseln Sie in die **Fussnotenansicht**.
3. Im Fuss- oder Endnotenfenster stellen Sie den Wert im dort sichtbaren Kombinationsfeld auf den Wert **Fussnotentrennlinie**.
4. Nehmen Sie die gewünschten Formatänderungen vor oder löschen Sie die Trennlinie nachdem Sie diese vorgängig markiert haben mit der Löschtaste.
5. Falls Ihnen die Formatierung nicht auf Anhieb gelingt, können Sie die Trennlinie mit der Schaltfläche ***Zurücksetzen*** wieder in den ursprünglichen Zustand versetzen.
6. Schliessen Sie den Dialog mit der Schaltfläche ***Schliessen*** ab.

Links zu diesem Thema

Microsoft

Deutsch

► Wie stelle ich die lange Fußnotentrennlinie ein?

Englisch

► WD97: How to Remove Footnote/Endnote Separators (Lines)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/FNtl.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich Symbole suchen und ersetzen?

◄ Zurück zum Hauptindex

Problemstellung

> In einem Dokument habe ich viele Sonderzeichen, welche ich über den Dialog *Einfügen - Symbol...* eingebracht habe. Ich sollte nun nach diesen Sonderzeichen suchen lassen und Sie ersetzen. Über den Suchen/Ersetzen-Dialog bringe ich dies nicht Zustande und zwar auch dann nicht, wenn ich die Option *Erweitern - Sonstiges* bemühe.

Lösung

Ihre Beobachtung ist richtig, es ist über den regulären Suchen/Ersetzen-Dialog wirklich nicht möglich z.B. Sonderzeichen aus der Symbolschrift *Wingdings* zu ersetzen.

Microsoft liefert allerdings ein Makro aus, welches einen Dialog enthält mit dem es möglich ist, dies zu bewerkstelligen. Sie finden dieses Makro in der Dokumentenvorlage namens **Makro80.dot**. Diese Vorlage finden Sie auf Ihrer **Office 97**-CD.

Word2000 Benutzer und Benutzerinnen sollten nach einer Dokumentenvorlage namens **Macros9.dot** Ausschau halten. Lassen Sie zuerst auf Ihrer Office2000-CD nach dieser Vorlage suchen. Wenn Sie dieses Produkt zu einem frühen Zeitpunkt erworben haben, ist es aber sehr wohl möglich, dass Sie die Vorlage dort nicht finden. In diesem Falle sollten Sie die Vorlage vom deutschen Server von Microsoft herunterladen. Sie finden den dazu nötigen Link am Schluss dieses Kapitels.

Suchen/Ersetzen-Dialog für Symbole verfügbar machen

Damit dieser Dialog Ihnen jederzeit zur Verfügung steht, sollten Sie wie folgt vorgehen:

1. **Öffnen** (Windows-Explorer - Rechtsklick - Kontextmenü - Öffnen) Sie die Vorlage Makro80.dot **Word 97** bzw. Macros9.dot **Word2000**.
2. Vom Word-Hauptmenü wählen Sie den Dialog ***Extras - Vorlagen und Add-Ins... - Organisieren...***.
3. Wechseln Sie im erscheinenden Dialog in den Reiter ***Makros*** bzw. Makroprojektelemente.
4. **Kopieren** Sie die beiden Komponenten **FindSymbolStart** und **frmFindSymbol** in die **Normal.dot**, welche sich lokal auf Ihrer Festplatte befindet.
5. Beenden Sie den Dialog mit der Schaltfläche ***Schliessen***.
6. Falls Sie aufgefordert werden, die Dokumentenvorlage **Normal.dot** zu speichern. kommen Sie dieser Aufforderung nach.

Suchen/Ersetzen-Dialog für Symbole aufrufen

Über den Dialog ***Extras - Makro - Makros... - FindSymbol - Ausführen*** können Sie diese Routine nun aufrufen. Der Rest ist dialoggeführt und eigentlich selbsterklärend.

Links zu diesem Thema

Microsoft

Deutsch

► Microsoft Word2000-Zusatzmakros: Macros9.dot

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Symbolse.htm
René Probst, Oktober 1999

Wie kann ich Etiketten mit einer Grafik (Logo) versehen?

◄ Zurück zum Hauptindex

Problem

> Ich erstelle Etiketten über den Word-Dialog *Extras - Umschläge und Etiketten - Etiketten* bzw. über *Extras - Seriendruck... - Adressetiketten....* Nun möchte ich das Etikett zusätzlich zum Text mit einem Logo schmücken. Der Dialog verwehrt mir aber die Möglichkeit über Kopieren - Einfügen eine Grafik einzubringen. Wie muss ich vorgehen?

Lösung

Grafik im Kopf des Etiketts positionieren

Um die Grafik mit dem Namen **"C:\Eigene Dateien\Mein Logo.jpg"** oberhalb des Textes, zu positionieren, gehen Sie wie folgt vor:

1. Skalieren Sie das Bild in einem Bildbearbeitungsprogramm (z.B. Photo Editor, welches im Office 97 Paket enthalten ist) zu einer geeigneten Grösse.
2. Vom Word-Hauptmenü wählen Sie den Dialog ***Umschläge und Etiketten*** und wechseln zum Reiter **Etiketten**.
3. Schreiben Sie im dafür vorgesehenen Fenster Ihren Text (Adressat).
4. Setzen Sie die Einfügemarke ganz an den Beginn des Textes und fügen Sie dort einen Zeilenumbruch ein.
5. Setzen Sie die Einfügemarke in diese neu erstellte Zeile.
6. Betätigen Sie die Tastenkombination **[Strg + F9]**; damit wird ein geschweiftes Klammerpaar generiert.
7. Setzen Sie die Einfügemarke zwischen diese beiden geschweiften Klammern.
8. Schreiben Sie nun zwischen den Klammern folgenden Text (Code)
   EinfügenGrafik "C:\\Eigene Dateien\\MeinLogo.jpg" für **Word 97** bzw.

   IncludePicture "C:\\Eigene Dateien\\MeinLogo.jpg" für **Word2000**.

   Achten Sie dabei darauf, dass der Pfad in **Gänsefüsschen** stehen muss und dass die ***backslashes (\\)*** **verdoppelt** werden müssen.
9. Betätigen Sie nun die Schaltfläche ***Neues Dokument***.
10. Markieren Sie nun das gesamte neu generierte Dokument mit der Tastenkombination **[Strg+A]**.
11. Betätigen Sie abschliessend die Taste **F9** um die Feldinhalte zu aktualisieren.

Grafik links des Adressaten positionieren

Um das Logo links des Textes, zu positionieren, gehen Sie wie folgt vor:

1. Vom Word-Hauptmenü wählen Sie den Dialog ***Umschläge und Etiketten...***. Anschliessend wechseln Sie zum Reiter **Etiketten**.
2. Schreiben Sie im dafür vorgesehenen Fenster Ihren Text (Adressat).
3. Markieren Sie den gesamten Text im Fenster und führen Sie mit der Maus einen Rechtsklick auf den markierten Text aus.
4. Im nun erscheinenden Kontextmenü wählen Sie **Absatz**.
5. Im erscheinenden Dialog stellen Sie unter **Einzug (links)** den erforderlichen Platz ein, welches das Logo einnehmen wird, also z.B. 3 cm. Schliessen Sie dann diesen Dialog mit ***Ok*** ab.
6. Betätigen Sie nun die Schaltfläche ***Neues Dokument***.
7. Wechseln Sie im neu generierten Dokument in die **Seitenlayoutansicht**.
8. Lassen Sie sich über ***Extras - Anpassen... - Symbolleisten*** die Symbolleiste mit dem Namen **Zeichnen** anzeigen.
9. Wählen Sie auf der Symbolleiste *Zeichnen* die Schaltfläche **Textfeld** und zeichnen Sie mit der Maus im ersten Etikett ein Textfeld und zwar an der Stelle an der die Grafik erscheinen soll.
10. Als nächstes gilt es, den Rahmen des Textfeldes zum Verschwinden zu bringen. Markieren Sie dafür das Textfeld am Rand, wählen Sie vom Hauptmenü ***Format - Textfeld...***, im Register **Farben und Linien** dieses Dialoges wechseln Sie die Einstellung im Auswahlfeld *Farbe:* auf **Keine Farbe**.
11. Platzieren Sie nun die Einfügemarke in das Textfeld und wählen Sie vom Hauptmenü den Dialog ***Einfügen - Grafik - Aus Datei...***.
12. Wechseln Sie zum Verzeichnis in dem sich die Grafik befindet und wählen Sie dann das Logo aus. Betätigen Sie zum Schluss die Schaltfläche ***Einfügen***.
13. Markieren Sie nun die Grafik und kopieren Sie diese mit der Tastenkombination **[Strg + C]** in die Zwischenablage.
14. Nun erstellen Sie mit der Tastenkombination **[Strg + V]** ein Kopie des Textfeldes welche die Grafik enthält. Diese Kopie des Logos wird sich über das erste Logo legen.
15. Ziehen Sie nun mit der Maus die Kopie des Logos in das zweite Etikett und platzieren Sie es dort korrekt.
16. Wiederholen Sie nun die letzen zwei Schritte für jedes weitere Etikett auf dem Formular.

Anmerkungen

- Wenn Sie die erste Methode gewählt haben, vergessen Sie bitte nicht, das Sie im Pfadnamen die Schrägstriche verdoppeln müssen. Beachten Sie auch, dass Sie in **Word2000** nur englischsprachige Feldnamen benützen können.
- Wenn Sie die zweite Methode gewählt haben ist es sinnvoller die Grafiken mit dem Dokument zu **Verknüpfen** statt sie einzubetten. Damit verhindern Sie, dass das Etikettformular sich unnötig aufgebläht.
- Beachten Sie bei der zweiten Methode ebenfalls, dass Sie bei einem Textfeld in Kontextmenü einstellen können, wie der Textfluss des Textes um das Textfeld aussehen soll. Diese Einstellung kann die Darstellung des Etiketts beeinflussen.
- Falls Sie Adressetiketten über die **Seriendruckfunktion** drucken möchten, können Sie grundsätzlich das gleiche Vorgehen verwenden.

Links zu diesem Thema

Microsoft

Deutsch

► **WD: Hinzufügen Grafik oder Logo zu jedem Etikett auf Seite**

Englisch

► **WD2000: How to Add a Graphic or Logo to Every Label**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/ELogo.htm

René Probst, November 1999

Änderungen

- --- nil ---

Wie kann ich in MS Word eine Datei unter einem fremdem Format öffnen oder abspeichern?

Die Umlaute in einem DOS-Text werden beim Öffnen unter Word falsch dargestellt!

Wie kann ich eine Vielzahl von Dokumenten in ein anders Format bringen?

Zusätzliche Text- und Grafikfilter von Microsoft

Text- und Grafikformate, für welche Microsoft keine Filter zur Verfügung stellt

Aus Word-Dokumenten PDF-Dateien erstellen

PDF-Dateien oder Teile davon ins Word-Format übernehmen

Dateiformate auf Grund ihrer Dateierweiterung identifizieren

VBA-technischer Umgang mit internen und externen Textkonverter

◄ Zurück zum Hauptindex

Diese Seite

▼ Was ist ein Konverter bzw. ein Filter?
▼ Die Methoden zur Darstellung von formatiertem Text
▼ Potential und Grenzen beim Konvertieren von formatiertem Text
▼ Datei explizit unter einem bestimmten Format Öffnen
▼ Datei explizit unter einem bestimmten Format speichern
▼ MS-DOS-Datei mit MS Word öffnen
▼ Batch-Konvertierung von Dateien (Stapelverarbeitung)
▼ Zusätzliche, aktuelle Text- und Grafikfilter für Office 97 und Office2000
▼ Verbreitete Text- und Grafikformate für welche Microsoft keine Filter bzw. Konverter bereit stellt
▼ Aus MS Word-Dokumenten PDF-Dateien erzeugen
▼ PDF-Dateien oder Teile davon ins Word-Format übernehmen
▼ Dateiformat auf Grund ihrer Dateierweiterung identifizieren
▼ VBA-technischer Umgang mit externen Word-Konvertern
▼ Links zu diesem Thema

Was ist ein Konverter bzw. ein Filter?

▲ Zurück zum Anfang

Der Begriff

Die Begriffe *Konverter* bzw. *Filter* werden in der Praxis meist synonym verwendet. Streng genommen, ist ein Konverter eher ein Stück Software, welches dem Export einer Datei in ein anderes Format ermöglicht, wogegen der Filter ein Stück Software darstellt, um eine Datei, welche ein fremdes Format besitzt zu importieren. Auch in diesem Skript, werden die beiden Begriffe synonym verwendet.

Damit ist schon festgehalten, dass die Konvertierung in beide Richtungen laufen kann. Gewisse Filter können nur Importieren, andere wiederum (dann eher Konverter genannt) können nur Exportieren und auch die Kombination von beidem ist durchaus gängig.

Für die Praxis heisst dies nun, dass folgende Konstellation auftreten können:

- Ein Filter für das gewünschte Format ist nicht verfügbar. Die Datei kann unter Word werde importiert noch exportiert werden.
- Es gibt einen Import- aber keinen Export-Filter
- Es gibt einen Export aber keinen Import-Filter
- Es gibt beides, einen Import- und einen Export-Filter
- Es gibt einen Import-/Export-Filter in einem

Text- und Grafikfilter

Auf einer ganz anderen Ebene wird auch zwischen **Text**- und **Grafikfilter** (bzw. Konverter) unterschieden. Dieses Skript geht in erster Linie auf Textkonverter und nicht auf Grafikfilter ein.

Soviel sei aber zu Grafikfiltern gesagt. Die zahlreichen Filter für Grafiken sind ausschliesslich für den Import bestimmt, ermöglichen Ihnen also Grafiken verschiedenster Formate in Ihrem Dokument einzubinden. MS Word kann keine Grafikdateien erstellen.

Die Methoden zur Darstellung von formatiertem Text

▲ Zurück zum Anfang

Nochmals auf einer anderen Ebene kann man drei Methoden unterscheiden, wie formatierter Text elektronisch dargestellt werden kann. Im einzelnen sind das folgende Methoden:

- **Tagging**: Auch unter *markup language* bekannt. Dabei werden Textsequenzen, welche eine gewisse Formatierung erhalten sollen vom Autor mit einer der Formatierung dienenden Kopf- und Schwanzinformation versehen. Die Texte werden also eigentlich nicht formatiert, sondern **codiert**. **HTML** ist ein typischer Vertreter dieser Kategorie. In den 80er Jahren, ist diese sehr zuverlässige Art, Text zu formatieren leider etwas in Vergessenheit geraten. Trotz eindeutigen Warnungen von Fachkreisen hat sich die Mehrheit der Benutzer auf *WYSIWYG*-Anwendungen (*What You See Is What You Get*) gestürzt. Mit dem Aufkommen des **Internets**, wo offene und stabile Strukturen besonders gefragt sind, hat diese Methode eine Renaissance erlebt. Diese Art von Textdarstellung (Codierung) ist **transparent** und ziemlich problemlos **erlernbar**. Die Codierung ist im Problemfall **analysierbar**. Dateien, welche so erstellt wurden, sind **vor Defekten weitgehend immun**. Dateikorruption ist so gut wie ausgeschlossen.

- **Formate:** Bei dieser Art Text zu formatieren, sind komplexe **Strukturen** und **Gefässe** vorgesehen, welche sowohl den Text als auch die Formatierung strukturiert aufnehmen können. Diese Architektur wird vor allem bei *WYSIWYG*-Anwendungen eingesetzt. **MS Word**, StarWriter und WordPerfect sind dafür typische Vertreter. Meist sind diese Formate **nicht herstellerunabhängig**, sondern properitär. Diese Art von Darstellung ist **nicht transparent** und nicht erlernbar. Eine Analyse im Problemfall müsste mit unter auf binärer Ebene geschehen und dies ist in der Praxis so gut wie unmöglich. Dateien, welche so erstellt wurden, sind **anfällig für Defekte**. Dateikorruption ist in der Praxis leider eine häufige Erscheinung.

- **PostScript** ähnliche Dateien. Diese Art von Dateien liegen schon fast so vor, wie ein Drucker den Datenstrom erwartet um seinen Job zu tun. Ein sehr populärer Vertreter in dieser Kategorie ist **PDF** (*Portable Document Format*) von **Adobe™**. Dokumente, welche dieser Architektur folgen sind plattformunabhängig, **layoutgetreu** (d. h. sie sehen auf jedem Rechner und im Ausdruck unabhängig vom Gerät haargenau gleich aus). Dateien dieser Klasse bieten auch einen gewissen **Schutz für Authentizität** können sie doch nachträglich nicht mehr verändert werden und nur sehr schwer in ein anderes Format überführt werden, welches eine Veränderung dann ermöglichen würde. Diese Dateien stellen also ein Endprodukt dar. Erstellt werden Sie nicht direkt im PDF-Format, sondern werden z.B. ex MS Word oder HTML erzeugt. Da an diesen Dateien nicht gearbeitet wird, sind sie vor Dateikorruption sehr sicher.

Potential und Grenzen beim Konvertieren von formatiertem Text

▲ Zurück zum Anfang

Es stellt sich nun die Frage, wie Dateien mit formatiertem Text, welche intern auf einem anderen Prinzip beruhen ineinander überführt, also konvertiert werden können. Im Zusammenhang mit dieser Frage lassen sich folgende Grundsätze festhalten.

- Die Übername von kodierten Dateien (also z.B. HTML) nach einer Datei, welche formatorientiert ist, geling meistens gut und detailgetreu.

- Das Umgekehrte, nämlich formatierten Text in kodierten Text über zu führen, ist hoch problematisch. Meistens enthält das Resultat viel Müll, dessen Volumen die eigentliche Substanz bei weitem übertrifft. Damit geht aber viel verloren, was *markup languages* im positiven Sinne auszeichnet. Die Transparenz, Analysierbarkeit und das Offene (also nicht Proprietäre) geht zu einem grossen Teil vor die Hunde. Der HTML-Export-Filter von MS Word2000 ist dafür ein abschreckendes Beispiel.

- Die Übernahme sowohl von kodiertem Text, wie auch von Textdateien, welche auf einem Format beruhen, zu einem PostScript-ähnliches Format ist selbstverständlich gewährleistet, anders würde diese Architektur ja keinen Sinn machen. Bedenken Sie aber, dass dazu zusätzliche Software benötigt wird (z. B. *Adobe Acrobat Writer*) und die ist in der Regel nicht billig.

- Der umgekehrte Weg, nämlich PostScript-ähnliche Dateien, auf ihr ursprüngliches Format (oder Kodierung) zurückzuführen ist in der Regel versperrt. Das hat einerseits technische Gründe (die Umkehrung ist nicht eindeutig), anderseits ist es von den Strategen auch so gewollt. Es muss ja wohl einen Weg geben, um auch auf elektronischem Wege etwas zu publizieren, von dem sowohl der Autor wie auch der Leser Gewähr hat, dass diese Publikation authentisch bleibt und nicht von irgendwelchen Personen unautorisiert und ohne Mitwissen der Betroffenen, verändert, verfälscht, missbraucht oder entwenden werden kann. Das kann auch als Service an den Leser gesehen werden, braucht doch auch er in vielen Fällen die Gewissheit, dass ihm ein authentisches Dokument vorliegt. Alles andere würde verunsichern. Schliesslich sei angeführt, dass geistiges Eigentum immer noch geschützt ist. Bei Zuwiderhandlung sieht unser demokratisch gestaltetes Gesetzt, empfindliche Strafen vor.

Datei explizit unter einem bestimmten Format Öffnen

▲ Zurück zum Anfang

Mittels Word-Dialog

1. Rufen Sie den Dialog ***Extras - Optionen...*** auf.
2. Wechsel Sie in das Register ***Allgemein***.
3. Aktivieren Sie die Option **Konvertierung beim Öffnen bestätigen**.
4. Schliessen Sie den Dialog mit ***Ok*** ab. Word wird sich später an diese Einstellung erinnern.
5. Öffnen Sie nun die fragliche Datei über den Dialog ***Datei - Öffnen***.
6. Word fragt bei Ihnen jetzt nach, in welchem Format die Datei zu öffnen ist. Wählen Sie von der Liste das gewünschte Format.

Anmerkungen

- Wenn die besagte Option ausgeschaltet ist, versucht Word selbst zu eruieren, welches Format am besten auf die Datei passen könnte. Das ist die Standardeinstellung in MS Word.
- Die Anzahl der Möglichkeiten, welche Ihnen Word anbietet hängt davon ab, welche Textkonverter Sie auf dem Rechner installiert haben. Dabei spricht man von Word-integrierten Filter (z.B. RTF), Textfilter, welche bei einer standardmässigen Installation von MS Office installiert werden, Filter welche Sie allenfalls bei einer benutzerdefinierten Installation von MS Office von der CD nachinstallieren können und letztlich Filter, welche im Konverter-Pack angeboten werden, welches Sie bei Microsoft herunter laden können. Die Downloadadresse für dieses Konverter-Pack finden Sie am Schluss dieses Skriptes.

Mittels VBA

Um eine Datei mit VBA unter einem bestimmten Format zu öffnen, können Sie in vielen Fällen auf **Word-integrierte Konstanten** zurückgreifen, welche sich im Objektkatalog von MS Word finden lassen. Sie können aber nicht davon ausgehen, dass dies für jedes Format zutrifft. Bedenken Sie, dass die Installation eines zusätzlichen Konverters nicht bewirkt, dass diese Konstanten dementsprechend in MS Word ergänzt würden. In diesem Falle sind Sie auf eine dynamischere Methode angewiesen. Dabei muss die Liste der verfügbaren Konverter ausgelesen werden. Damit Sie nicht direkt auf die Registrierdatenbank zugreifen müssen, sieht Word-VBA hier eine handlicher Methode vor.

Unter Einsatz der Word-integrierten Konstanten

Sollten Sie das gewünschte Format in der unten gezeigten Tabelle finden, dann lautet die Syntax um eine Datei unter einem bestimmten Format zu öffnen wie folgt:

```
Documents.Open FileName:="C:\Eigene Dateien\Test.rtf", Format:=wdOpenFormatRTF
```

(Öffnen einer Datei, explizit im RTF-Format)

Tabelle der integrierten Format-ID's

Öffnen	Speichern	Bemerkung
wdOpenFormatAllWord	---	---
wdOpenFormatAuto	---	Standard zum Öffnen
wdOpenFormatDocument	wdFormatDocument	---
---	wdFormatDOSText	---
---	wdFormatDOSTextLineBreaks	---
wdOpenFormatEncodedText	wdFormatEncodedText	---
---	wdFormatHTML	---
wdOpenFormatRTF	wdFormatRTF	---
wdOpenFormatTemplate	wdFormatTemplate	---
wdOpenFormatText	wdFormatText	---
---	wdFormatTextLineBreaks	---
wdOpenFormatWebPages	---	**Word2000** spezifisch
wdOpenFormatUnicodeText	wdFormatUnicodeText	---

Falls keine entsprechende Word-integrierten Konstanten vorhanden ist
Sollte es für das von Ihnen gesuchte Format keine Konstante finden lassen, dann lesen Sie das Kapitel *Festellen welche Konverter auf dem System vorhanden sind* etwas weiter unten in diesem Skript. Falls Sie eine **MS-DOS-Datei** als solches code-technisch unter MS Word öffnen möchten, dann lesen Sie bitte das Kapitel *MS-DOS-Datei mit MS Word öffnen* in diesem Skript.

Datei explizit unter einem bestimmten Format speichern

▲ Zurück zum Anfang

Mittels Word-Dialog

1. Rufen Sie vom Word-Hauptmenü den ***Dialog Datei - Speichern unter...*** auf.
2. Stellen Sie im Kombinationsfeld mit der Bezeichnung **Dateityp:** das gewünschte Format ein.
3. Klicken Sie auf die Schaltfläche **Speichern**.
4. Falls Word eine Warnung anzeigt, bestätigen Sie Ihre Wahl.

Anmerkung

- Die Anzahl der Möglichkeiten, welche Ihnen Word anbietet hängt davon ab, welche Textkonverter Sie auf dem Rechner installiert haben. Dabei spricht man von Word-integrierten Filter (z.B. RTF), Textfilter, welche bei einer standardmässigen Installation von MS Office installiert werden, Filter welche Sie allenfalls bei einer benutzerdefinierten Installation von MS Office von der CD nachinstallieren können und letztlich Filter, welche im Konverter-Pack angeboten werden, welches Sie bei Microsoft herunter laden können. Die Downloadadresse für dieses Konverter-Pack finden Sie am Schluss dieses Skriptes.

MS-DOS-Datei mit MS Word öffnen

▲ Zurück zum Anfang

Die Stolperfalle hier ist, dass Word nicht zwingend erkennen kann, dass die Datei unter MS DOS entstanden ist, vielmehr geht die Anwendung davon aus, dass es sich um eine Textdatei handelt, welche unter Windows entstanden ist.

Text wird unter MS-DOS teilweise anders codiert als unter Windows. Der ANSI-Code für die unteren 127 Zeichen in der Zeichentabelle deckt sich zwar noch mit dem ASCII-Code, die oberen 127 Zeichen des ANSI-Codes sind aber länderspezifisch und decken sich nicht mit dem erweiterten ASCII-Code. Dies wirkt sich vor allem in **nicht** angelsächsischen Sprachen aus. Was die Deutsche Sprache betrifft, sind vor allem Umlaute betroffen, allerdings auch einige Sonderzeichen.

Das ist aber überhaupt kein Grund zur Verzweiflung, denn Word kann sehr wohl eine MS-DOS Datei richtig interpretieren. Allerdings will es davon unterrichtet sein. Gehen Sie konkret wie folgt vor:

1. Rufen Sie den Dialog ***Extras - Optionen...*** auf.
2. Wechsel Sie in das Register ***Allgemein***.
3. Aktivieren Sie die Option **Konvertierung beim Öffnen bestätigen**.
4. Schliessen Sie den Dialog mit ***Ok*** ab. Word wird sich später an diese Einstellung erinnern.
5. Öffnen Sie nun die fragliche Datei über den Dialog ***Datei - Öffnen***.
6. Word fragt bei Ihnen jetzt nach, in welchem Format die Datei zu öffnen ist. Wählen Sie von der Liste **MS DOS-Text**.

MS DOS-Datei mit Word-VBA öffnen

Das korrekte Öffnen (also mit Konvertierung) einer MS-DOS Datei mittels Word-VBA ist schon viel eher ein Grund zum verzweifeln, hat Microsoft doch in diesem Bereich eindeutig etwas verschlampt. Eine word-integrierte *OpenFormatID* dazu gibt es nicht und um einen externer Konverter (was die dynamische Ermittlung der Format-ID zulassen würde) handelt es sich auch nicht. Sie können zwar, wenn Sie die Format-ID dynamisch ermitteln, wie dies in diesem Skript noch besprochen wird, den Filter mit dem Namen *MS-DOS-Text mit Layout* applizieren. Damit erreichen Sie einen Teilerfolg, die Umlaute, z.B. werden dann nämlich richtig umgesetzt nicht aber die Zeilenumbrüche.

☹ Microsoft hat es also verschlampt uns eine Methode zur Verfügung zu stellen, welche unter Einsatz von Word-VBA ermöglichen würde, eine MS-DOS-Datei ohne Intervention des Benutzers zu öffnen.

Dank einem ***WordBasic*-Befehl** ergibt sich eine interessante Umgehung dieses Problems, welche sich problemlos mit den neueren Word-Versionen verträgt. Dabei wird die DOS-Datei Zeile für Zeile mit VB eingelesen und unter Einsatz eines *WordBasic*-Befehles nach ANSI konvertiert. Der *WordBasic*-Befehl, der dies kann nennt sich **DOSToWin** (als hätte Microsoft ANSI erfunden und als wäre ANSI ein Synonym zu Windows **kopfschüttelimquadrat*) Die ganze Routine sieht dann wie folgt aus:

```
Private Const Pfad = "C:\DOSText.asc"

Sub DOSnachANSI()
  Dim nDoc As Document, oRange As Range
  Set nDoc = Documents.Add
  Set oRange = nDoc.Content
  Open Pfad For Input As #1
  While Not EOF(1)
    Line Input #1, X
    X = WordBasic.[DOSToWin$](X)
    oRange.InsertAfter X & vbCrLf
  Wend
  Close #1
End Sub
```

(Unter Einsatz von *WordBasic* wird DOS-Text nach ANSI konvertiert.)

Batch-Konvertierung von Dateien (Stapelverarbeitung)

▲ Zurück zum Anfang

Sowohl im Lieferungsumfang von Office 97 wie auch von Office2000 ist ein *Wizard* enthalten, welcher es Ihnen dialoggeführt erlaubt, ein ganzes Verzeichnis mit Dateien in ein anderes Format überzuführen. Damit erübrigt sich die mühselige Arbeit ein Dokument nach dem anderen zu Öffnen und dieses Dokument dann unter einem anderen Format zu abzuspeichern. Das Wizard nennt sich in **Office 97 Konvert8.wiz** und in **Office2000 Batch Conversion Wizard.wiz**. Sie finden diese Dateien entweder auf Ihrer Festplatte oder aber auf Ihrer Office-CD und zwar auf der ersten in der Reihe, sollte Ihre Office-Ausgabe denn auf mehrere Scheiben aufgeteilt sein.

Zu mindest, was Office2000 betrifft, können Sie diesen Assistenten auch über den Office-Setup installieren, mit dem Vorteil, dass Sie dieses Werkzeug dann auch über ***Neues Office-Dokument - Sonstige Dokumente - Konvertier-Assistent*** aufrufen können. Gehen Sie als Anwender von Office2000 konkret wie folgt vor:

1. Schmeissen Sie Ihre Office2000-CD ins Laufwerk. Sollten Sie mehrere Scheiben besitzen - es ist die erste der Reihe.
2. Klicken Sie ***Start - Einstellungen - Systemsteuerung - Software***.
3. Suchen Sie in der erscheinenden Liste nach **Microsoft Office2000...** und markieren Sie diesen Einträg.
4. Klicken Sie auf **Hinzufügen/Entfernen**.
5. Im Wartungsmodus angekommen klicken Sie auf **Features Hinzufügen/Entfernen**.
6. Erweitern Sie sie die Auswahl indem Sie zuerst auf **Microsoft Word für Windows** und dann auf **Office-Assistenten** klicken.
7. Klicken Sie nun auf **Weitere Assistenten** und wählen Sie im Kontextmenü **Vom Arbeitsplatz starten**.
8. Betätigen Sie abschliessend den Schaltknopf J**etzt aktualisieren**.

Sollten Sie sich damit begnügt haben, die besagte Datei (*Wizard*) zu kopieren können Sie das *Wizard* starten, wenn Sie einen Doppelklick auf die genannte Datei ausführen. Sollten Sie den Assistenten ordnungsgemäss installiert haben, können Sie das Werkzeug auch über *Neues Office-Dokument* starten. Befolgen Sie anschliessend die Anweisungen des Konvertier-Assistenten.

Zusätzliche, aktuelle Text- und Grafikfilter für Office 97 und Office2000

▲ Zurück zum Anfang

Es wurde hier schon erwähnt, das man die Filter bzw. Konverter, welche Microsoft zur Verfügung stellt in folgende Kategorien unterteilen kann:

- Filter welche direkt in MS Word eingebaut sind (z.B. RTF)
- Zusätzliche Filter, welche bei einer standardmässigen Installation von MS Office installiert werden
- Zusätzliche Filter welche über den Office-Setup (*Einstellungen - Systemsteuerung - Software Microsoft Office...*) nachinstalliert werden können
- Zusätzlich (aktuelle) Filter, welche Sie über das Internet von Microsoft herunter laden können

Es ist sicher kundenfreundlich, dass Microsoft nun ein Paket zur Verfügung stellt, welches sämtliche Text- und Grafikfilter beinhaltet, die für MS Word bzw. MS Office Microsoft-seitig zur Verfügung stehen. Unabhängig davon, ob sie Word 6.x, Word 95 (alias Word 7.0), Word 97 oder Word2000 einsetzen, Sie können das selbe Paket mit den aktuellsten Filter verwenden.

Die Tatsache, dass es Microsoft gelingt, gerade im Grafikbereich so viele Filter zur Verfügung zu stellen, liegt natürlich auch daran, dass diese Formate oft herstellerunabhängig und sehr gut dokumentiert sind. Leider orientiert sich Microsoft selbst eher selten so vorbildlich. Viele Formate, welche von Microsoft entwickelt werden sind proprietär nicht mit anderen Herstellern abgesprochen. Auch was die Dokumentation betrifft, hapert es bei Microsoft. Macht man Formate öffentlich, dann ist das übrigens auch ein Zeichen von Selbstvertrauen. Jede technisch interessierte Person, kann sich dann selbst eine Meinung bilden, wie intelligent und konsistent so ein Format ist. Es mag natürlich einen kommerziellen bzw. pekuniärer Anreiz haben, proprietäre Formate zu generieren

Der Download ist allerdings etwas gross geworden, umfasst dieser doch knapp **13MB** Daten. Nach dem Download können Sie das Paket mittels Mausklick installieren und dann auswählen, welche Text- und Grafikfilter auf Ihrem Rechner installiert werden sollen.

Konverter sind übrigens so gebaut, dass diese völlig unabhängig von der Benutzeroberfläche arbeiten, Deshalb gibt es auch keinen deutschsprachige Version des Konverter-Packs und was auch nicht nötig ist. Den Download von Microsoft können Sie mit einem Klick auf den folgenden Link initialisieren.

► **Konverter-Pack von Microsoft (Download 13MB)**

Text- und Grafikfilter, welche im Konverterpack enthalten sind

Textfilter
Das Paket enthält folgende Text- und Datenbankfilter:

Für Word2000

- **Ami** Pro 3.0
- Filter für **dBase** Datenbanken
- Filter für **FoxPro** Datenbanken
- MS **Word** 3.x - 5.x für **MS-DOS**
- MS **Works** 3.0
- **RFT-DCA**
- MS **Write**
- **WordPerfect** 4.x
- **WordStar**

Für Word 97

- **Ami** Pro 3.0
- Filter für **dBase** Datenbanken
- Filter für **FoxPro** Datenbanken
- **HTML**
- **Lotus 1-2-3**
- **Excel**
- MS **Word 6.x** und MS **Word 95**
- **MS Word** 3.x - 5.x **für MS-DOS**
- MS **Word** 5.x **für Macintosh**
- MS **Works 3.0**
- MS **Works 95**
- **RFT-DCA**
- **Textdateien**
- MS **Write**
- **WordPerfect 4.x**
- **WordPerfect 5.x**
- **WordPerfect 6.x**
- **WordStar**

Word 95 (alias Word 7.0)

- **Ami** Pro 3.0
- Filter für **dBase** Datenbanken
- Filter für **FoxPro** Datenbanken
- **HTML**
- **Lotus 1-2-3**
- **Excel**
- **Word 97** und **2000**
- MS **Word** 3.x - 5.x für **MS-DOS**
- MS **Word** 5.x für **Macintosh**
- MS **Works 3.0**
- MS **Works 95**
- **RFT-DCA**
- **Textdateien**
- MS **Write**
- **WordPerfect 4.x**
- **WordPerfect 5.x**
- **WordPerfect 6.x**
- **WordStar**

Für Word 6.x

- **Ami** Pro 2.0
- Filter für **dBase** Datenbanken
- Filter für **FoxPro** Datenbanken
- **Lotus 1-2-3**
- **Excel**
- MS **Word 97** und **2000**
- MS **Word** 3.x - 5.x **für DOS**
- MS **Word** 5.x für **Macintosh**
- **RFT-DCA**
- **Textdateien**
- MS **Write**
- **WordPerfect 4.x**
- **WordPerfect 5.x**
- **WordStar**

Grafikfilter

Alle Word-Versionen (ab 32-Bit)

- Windows Bitmap (**BMP**)
- **GIF**
- JPEG (**JPG**)
- TIFF (**TIF**)
- Windows Meta File (**WMF**)
- **CDR** (CorelDraw Grafiken) Nur bis und mit CorelDraw Version 6.0!
- **CGM** (Computer Graphics Metafile)
- **DRW** (Micrografx Designer/Draw)
- **DXF** (AutoCAD)
- **EMF** (Windows Enhanced Metafile)
- **EPS** (Encapsulated PostScript)
- **FPX** (FlashPix)

Verbreitete Text- und Grafikformate für welche Microsoft keine Filter bzw. Konverter bereit stellt

▲ Zurück zum Anfang

Grafikfilter

☹ Nicht unterstützt sind **CorelDraw™** Grafiken **höher als Version 6.0**. Der Filter, welcher Microsoft noch zur Verfügung stellt, ist also für PC Begriffe uralt. Der Grund, weshalb Microsoft diesen Filter nicht hochzieht ist der, dass der kanadische Softwarehersteller *Corel* 1996 alle Rechte für *WordPerfect*™ erworben hat. WordPerfect stellt eine Konkurrenz zu MS Word dar. Microsoft hat dementsprechend reagiert. Sie können mit der Zwischenablage arbeiten, um diese Grafiken trotzdem zu transferieren. Die Qualität leidet allerdings darunter klar sichtbar. Der bessere Weg ist sicher, diese Grafikdateien unter *CorelDraw* in einem Word-kompatiblen Format abzuspeichern und sie dann in MS Word zu übernehmen. *CorelDraw* Grafikdateien laufen unter der Dateierweiterung ***.cdr**.

Textkonverter

☹ Microsoft stellt weder Import- noch Exportfilter für Dokumente, welche unter **StarWriter™** besser bekannt als **StarOffice™** erstellt wurden. Dieser Textprozessor bzw. dieses Office-Paket wird heute vom der Firma *Sun Microsystems* vertrieben, Rechte welche die Firma von *Star Division Corp* erworben hat. Der Grund, wieso Microsoft keinen Filter zur Verfügung stellt, ist Konkurrenzdenken. Liegen Ihnen solche Dokumente vor, dann haben Sie grundsätzlich drei Optionen. Sie können diese Datei mit *StarWriter* unter einem Format abspeichern, welches MS Word interpretieren kann. Benutzer von *StarWriter* rühmen den **MS Word Exportfilter**, welcher in *StarWriter* integriert ist. Alternativ können Sie die Datei auch in *StarWriter* im **RTF-Format** abspeichern, ein Format, welches MS Word problemlos und ohne Zusatz interpretieren kann.*StarWriter* Dateien laufen unter der Dateierweiterung ***.swd**.

☹ Seit der Textprozessor, welcher ursprünglich unter *Ami WordPro* vermarktet wurde zuerst an *Lotus* und dann an *IBM* übergegangen ist, werden die Filter für diesen Textprozessor von Microsoft nicht mehr hochgezogen. Die letzte Version, für welche es von Microsoft noch einen Filter gibt, heisst *Ami Pro 3.0*. Für Sämtliche **Lotus WordPro™** Versionen, gibt es **keinen** Filter von Microsoft. Grund für das Fehlen eines Konverters ist Konkurrenzdenken. Um solche Dateien in MS Word verfügbar zu machen, speichern Sie das Dokument in *Lotus WordPro* im *RTF*-Format oder im *WordPro 3.0* Format ab. MS Word kann *RTF* direkt interpretieren, für das *Ami Pro 3.0* Format ist im Konverter-Pack von Microsoft (den Link dazu finden Sie in diesem Skript) einen Filter vorgesehen. Sollten Sie keinen Zugang zu *Lotus WordPro* haben, dann können Sie sich einen Filter von einem Drittanbieter erwerben. Schauen Sie sich dazu den am Schluss dieses Skriptes eingeblendeten Link von Microsoft bezüglich *Lotus WordPro* an. Microsoft nennt in diesem Artikel auch Firmen, welche solche Filter vertreiben. Die letzte Alternative, welche sich Ihnen bietet, ist die *Lotus WordPro* Datei unter Word zu öffnen. Stellen Sie dann aber den Dateityp auf **Text aus beliebiger Textdatei wiederherstellen**. Sie müssen dann selber beurteilen, ob dieses Ergebnis für Sie brauchbar ist. Lous WordPro Dateien laufen unter der Dateierweiterung ***.lwp**.

Aus MS Word-Dokumenten PDF-Dateien erzeugen

▲ Zurück zum Anfang

Aus verständlichen Gründen nimmt der Wunsch stetig zu, z.B. aus Word-Dokumenten Dateien im PDF-Format zu erstellen. Die Ausbreitung des Internets hat diesen Wunsch noch massiv verstärkt. PDF speichert Dokumente, welche formatierten Text und direkt eingebettete Bilder enthalten können, in einem PostScript-ähnlichen Format. Werkzeuge, wie **Adobe Acrobat™,** können die an sich sehr aufgeblasenen PostScript-Dateien fast ohne Verlust an Qualität auf eine erträgliche Grösse komprimieren. Der Nutzen des Acrobat ***PDFWriters*** liegt also vor allem in der Leistung, welche er im Bezug auf die Komprimierung an den Tag legt, denn PostScript-Dateien lassen sich auch mit kostengünstiger Software erzeugen, oft sogar ganz gratis. Beim ***Adobe Acrobat Distiller*** kommt hinzu, dass dieser z.B. die Strukturen eines Word-Dokument aufbrechen und neu assemblieren (zusammensetzen) kann. Damit zeichnet sich dann auch das PDF-Abbild dadurch aus, dass Inhaltsverzeichnisse, Kommentare und u.a. auch funktionell vorhanden ist. Das will heissen, Inhaltsverzeichnisse sind mit Hyperlinks unterlegt, Kommentare *popen* auf und was MS Word halt sonst noch so zu bieten hat wird auch funktionell möglichst detailgetreu abgebildet.

Dass eine solche Software eine Preis hat, scheint mir verständlich. Es braucht enorm viel *know how* um z.B. ein Word-Dokument in diesem Sinne zu verarbeiten. Die Entwicklung dazu ist auch zweitaufwendig und riskant. Riskant deshalb, weil man daran scheitern kann (der Distiller macht dann auch prompt Fehler) und weil man der Entwicklung von Microsoft auf Gedeih oder Verderben immer hinten her rennen muss. Kommt hinzu, dass das Format von MS Word nicht sonderlich gut dokumentiert ist und dass man in manchen Fällen nicht darum herum kommt, die vielen BUGS, welche in MS Word ihr Unwesen treiben, nachzubilden.

Der Preis von ca. DM 600,00 für den *Adobe Acrobat 4.0*, dünkt mich dann aber vor allem für private Zwecke doch sehr überteuert. Knapp die Hälfte des Preises, wäre wohl eher zu rechtfertigen. Wirklich gute Alternativen sind allerdings rar. PDF Dateien zu erzeugen, ist allerdings kein Privileg von Adobe und Sie finden im Internet da und dort legale Alternativen dazu. Die Links dieser Angebote wechseln häufig und dienen mit unter vor allem auch dafür, Leute auf eine bestimmte Home Page zu locken. Für die meisten dieser Werkzeuge brauchen Sie einiges an Geduld und *know how*, bis da wirklich etwas zu laufen kommt.

Wie schon erwähnt, eine der Stärken von Adobe Acrobat liegt in der Fähigkeit zu komprimieren. Wenn Sie einen anderen Weg nehmen, um zu PDF-Dateien zu erzeugen, kommen Sie in der Regel an einem Kompromiss zwischen Qualität und Dateigrösse nicht herum. Sie können mit hoher Auflösung operieren, dann wächst die Dateigrösse einer quadratischen!!! Funktion folgend, oder Sie können die Auflösung tief wählen, dann leidet das Aussehen Ihres Werkes.

Somit dürfte die Marketingtaktik von Adobe wohl aufgehen. Man vertreibt den Acrobat Reader gratis, propagiert diesen an allen Ecken und Enden und schafft sich damit einen Markt für das PDF-Format. Die Lösung, um solche Dateien zu erzeugen hat man auch in der Tasche. Diese Lösung wiederum rückt man aber nur gegen viel gutes Geld heraus.

> ! Verwechseln Sie den Adobe Acrobat Reader nicht mit Adobe Acrobat. Der Reader erhalten Sie gratis, wogegen die Software von Adobe zur Erzeugung von PDF-Dateien ausschliesslich bei Ihrem Software-Händler für ca. DM 600, 00 zu haben ist.

Wenn Sie sich für Adobe Acrobat 4.0 entschieden haben, erwerben Sie sich eine Software, welche im wesentlichen folgendes kann

Acrobat PDFWriter: Es handelt sich um einen (virtuellen) Druckertreiber. Statt auf einem Drucker landet der Ausdruck direkt (also ohne Umweg Drucken nach Datei umleiten) in einer PDF-Datei. Diese Datei kann im Acrobat Reader betrachtet werden und dann auf Wunsch ausgedruckt werden. Über spezielle Funktionen, wie z.B. Inhaltsverzeichnisse, welche als Hyperlinks fungieren, verfügt ein solche Datei nicht. Die Komprimierungsrate ist gut aber nicht sehr gut. Die optische Qualität ist in der Regel gut, allerdings gelten qualitative Einschränkungen was Word-Tabellen etc. betrifft. Der Vorgang des Erzeugens ist schnell.

Acrobat PDF Distiller: Erlaubt es Ihnen z.B. aus einer Word-Datei eine PDF-Datei zu erzeugen. Die Struktur und die Elemente, welche im Word-Dokument enthalten sind, werden möglichst 1:1 ins PDF Format umgesetzt. Dies schliesst auch funktionelle Elemente mit ein. Die Umwandlung geschieht sehr detailgetreu. Die optische Qualität ist sehr gut. Die Komprimierungsrate erstaunlich. Der Vorgang kann bei grossen und komplex-strukturierten Dokumenten mehr als eine Kaffeepause in Anspruch nehmen. Ganz fehlerfrei geschieht die Konvertierung bei komplexen Dokumenten allerdings selten.

Acrobat Web Capture: Erlaubt es Ihnen eine Site im Internet herunter zu laden, wobei ein PDF-Dokument entsteht. Auf Wunsch kann dieses sehr praktische Werkzeug auch weiterführende Links selbständig aufgreifen und dies über mehrere Hierarchiestufen. Die einzelnen miteinander verhängten Web-Seiten werden dann im Inhaltsverzeichnis, welches automatisch generiert wird, hierarchisch korrekt aufgeführt. Somit können Sie die ganze Baumstruktur eines Web-Auftrittes mit sehr geringem Arbeitsaufwand aus dem Internet ziehen und offline verfügbar machen. Leider kann Adobe Acrobat CSS (Cascaded Style Sheets) nicht interpretieren. Deshalb wird die Formatierung von Sites, welches dieses verbreitete Hilfsmittel nutzen, nicht übernommen. Diese Web-Seiten erscheinen dann im HTML-Standardformat.

Nicht zuletzt haben Sie mit Adobe Acrobat ein komplettes Werkzeug in der Hand, welches es Ihnen ermöglicht, **komplette Dokumente**, welche alles enthalten also auch Grafiken und allenfalls Schriftarten, auf elektronischem Wege zu **verbreiten**. Der Acrobat Reader, ist für **alle PC-Plattformen** gratis und ohne spezielle Umtriebe erhältlich. Da Dokumente, welche im PDF-Format vertrieben werden, kaum veränderbar sind und kaum in andere Formate konvertiert werden können, haben Sie und der Leser eine ziemlich verlässliche **Garantie auf Authentizität**. Sie können übrigens festlegen, ob das Dokument gedruckt werden kann und ob Datenaustausch über die Zwischenablage möglich sein soll.

PDF-Dateien oder Teile davon ins Word-Format übernehmen

▲ Zurück zum Anfang

Erzeugt man ein PDF-Dokument dann sind die Optionen grundsätzlich mal so vorgegeben, dass das Dokument verändert, gedruckt und Teile des Textes bzw. Grafiken kopiert werden können. Der Autor bzw. die Autorin kann allerdings diese genannten Optionen einzeln oder alle samt ausschalten. Dabei handelt es sich in der Regel um einen bewussten Akt des Verfassers.

Auch wenn das Dokument für *Copy/Paste*-Operationen frei gegeben ist, dann bleibt beim Kopieren über die Zwischenablage ausschliesslich die Zeichenformatierung (teilweise) erhalten, nicht aber die Absatzformatierung. Es ist grundsätzlich möglich, Grafiken (eine um die andere) über die Zwischenablage auszutauschen, man berichtet aber, dass die Qualität der Bilder darunter sichtbar leidet. Auch die Möglichkeit des Kopierens von Grafiken bedingt, dass das Dokument für *Copy/Paste*-Operationen frei geben ist.

Vom technischen Standpunkt aus gesehen spricht vieles, allerdings nicht alles dafür, dass sich das PDF-"Format" für solche Operationen schlecht eignet. Ein PDF-Dokument stellt nun mal ein Endprodukt dar. Wenn es weiter vertrieben wird, dann soll dies ausschliesslich in der vom Verfasser oder von der Verfasserin authentisierten Form geschehen. Diesen Gedanken hatten die Erfinder von PDF bei der Entwicklung vor ihren Augen und haben die Technologie entsprechend ausgesucht. Der Schutz, ist zwar nicht ganz sicher und nicht unüberwindbar, aber es braucht doch etliches an Werkzeug und *know how* um ihn zu knacken. Auch wenn jemand über diese Mittel verfügt, ist es fraglich, ob ihm der zeitliche Aufwand, welcher er dazu benötigt, die Sache wert ist.

Ich denke auch, dass Adobe weiter an einem noch bessern Schutz arbeitet, wenn in diesem Bereich noch Potential vorhanden ist. Damit wird wohl das muntere Polizist- und Räuberspiel weitergehen, wobei Chaoten die Zeche verursachen und der Seriöse bzw. die Allgemeinheit diese Zeche bezahlt. Am Ende dürfte das so weit gehen, dass auch der wahre Urheber grosse Nachteile tragen muss, sollte er sich mal durch eine Verkettung und Missgeschicken selbst aus der Sache ausschliessen. Schlimmer an der Sache dünkt mich aber, dass auch die Leser von Publikationen im PDF-Format nicht ganz sicher sein können, ob ihnen ein authentisches Dokument vorliegt.

Wenn jemand ein PDF-Dokument von Drittpersonen in irgend einer Form weiterverarbeiten möchte, dann soll er genau das tun, was er im Falle eines Buch- oder einer Zeitungsartikels auch tun würde, nämlich beim Verfasser nachfragen, ob sein Ansinnen überhaupt genehm ist. Falls die Anfrage positiv beantwortet wird, kann der Urheber Ihnen entgegen kommen, ist er doch gewiss im Besitze einer Kopie der Publikation, welche nicht in PDF sondern in einem editierbaren Format vorliegt.

Beachten Sie bitte auch den Link zu einem Artikel von Microsoft, welcher einige Kniffe aufzeigt, um Textstellen und Grafiken aus einem PDF-Dokument z.B. nach Word zu übernehmen. Den Link zu diesem Skript finden Sie ganz am Ende dieser Seite.

Dateiformat auf Grund ihrer Dateierweiterung identifizieren

▲ Zurück zum Anfang

Im folgenden finden Sie eine Auflistung von verbreiteten Dateiendungen. Die Tabelle gibt Auskunft darüber, bei welcher Dateierweiterung Sie welches Format erwarten dürfen. Bitte beachten Sie, dass es bei diesen Dateierweiterungen lediglich um eine Konvention handelt, welche mal im Alleingang von Microsoft eingeführt wurde. Es handelt sich dabei um einen sehr pragmatischen Ansatz, welcher MS Windows entgegen kommt. Es ist selbstverständlich ein Kinderspiel z.B. eine Excel-Datei in eine Word-Datei umzubenennen. Dabei entsteht selbstverständlich keine Word-Datei. Im technischen Sinne sagt die Dateierweiterung also gar nichts über das Format aus.

Die Tabelle gibt auch darüber Auskunft, ob MS Word mit den aufgeführten Dateiformaten etwas anfangen kann und was für Filter allenfalls nachinstalliert werden können.

Wo es keinen direkten Weg gib, die Datei nach MS Word zu überführen bzw. in Word einzuführen, gehen Sie grundsätzlich wie folgt vor:

- Speichern Sie die Datei mit der Anwendung, aus welcher sie hervorgegangen ist, in ein Word-kompatibles Format ab. Benutzen Sie dazu z.B. das RTF-Format. Öffnen Sie dann die konvertierte Datei in MS Word.

- Alternativ, erwerben Sie sich bei einem Drittanbieter einen tauglichen Konverter für das fragliche Dateiformat. Drittanbieter, welche solche Werkzeuge anbieten, finden Sie in reichlicher Zahl im Internet. Selbstverständlich muss die Architektur des fremden Formates von einer Beschaffenheit sein, dass eine Konvertierung in das MS Word-Format technisch machbar ist. Über das Potential und die Grenzen von einer Dateikonvertierung wurde in diesem Skript bereits gesprochen. Bedenken Sie auch, dass die Entwicklung von Konverter eine sehr anspruchsvolle und zeitintensive Arbeit darstellt. Da sich Formate meistens weiter entwickeln und deren Marktanteil ständig wechselt, ist auch ein beträchtliches unternehmerisches Risiko im Spiel. Drittanbieter, welche über dieses Potential verfügen, werden sich in der Regel Ihr Produkt vergolden lassen. Ein Interesse, solche Filter gratis abzugeben, haben allenfalls nur jene beiden Firmen, welche direkt involviert sind. Der Kunde wiederum, ist meistens der Leidtragende, wenn er auf herstellerabhängige Formate setzt. MS Word ist leider so ein properitäres Format.

Alphabetische Auflistung von Dateierweiterungen wie sie unter Windows von Text- und Grafikdateien verwendet werden

Datei-Endung	Genre	Hersteller	Steht für	Filter in MS Word	Bemerkungen
123	Tabellenkalkulation	Lotus (heute Tochter von IBM)	---	Nur in älteren Word-Versionen	PC Steinzeit Uralt
apr apt	Datenbank	Lotus	Approach DB	Nein	---
Ans	Textdatei	Hersteller unabhängig	ANSI	Ja	Eigentlich Text, welcher in ANSI codiert ist (im Gegensatz zu ASCII-Dateien unter MS-DOS). Diese Dateierweiterung wird aber nicht immer strickt angewandt. Siehe dazu auch Textdateien (txt).
Asc	Textdatei	Hersteller unabhängig	ASCII	Ja	Eigentlich Text, welcher in ASCII (bzw. erweiterter ASCII-Code) codiert ist (im Gegensatz zu ANSI codierten Dateien unter Windows). Diese Dateierweiterung wird aber nicht immer strickt angewandt. Siehe dazu auch Textdateien (txt).
Bmp	Grafik	Hersteller unabhängig	Windows Bitmap	Ja	Grafikformat, welches detail- und farbgetreu ist, welches aber viel Speicherplatz braucht und sich schlecht skalieren lässt.
Cdr	Grafik	Corel	CorelDraw	Nur alte Versionen sind unterstützt	Nur bis und mit CorelDraw 6.0. Siehe dazu das Kapitel Verbreitete Text- und Grafikformate für welche Microsoft keine Filter bzw. Konverter bereit stellt
Cgm	Grafik	Hersteller unabhängig	Computer Graphics Metafile	Ja	Vektororientiertes Grafikformat.
Csv	Tabellen und Datenfelder	Hersteller unabhängig	Comma Separated Value	Ja	Es handelt sich um Dateien, welche Elemente enthalten, welche z. B. durch ein Komma (oft auch Semikolon) getrennt sind. Diese Trennzeichen ergeben dann Spalten, die Zeilenumbrüche Zeilen. Wird von Word in einem Dokument als reine Zeichenfolge übernommen. Als Seriendruckdatenquelle geeignet und korrekt interpretiert.
Dbc	Datenbank	Microsoft	Data Base Container	Ja	FoxPro Datenbanken Als Seriendruckdatenquelle geeignet.
Dbf	Datenbank	Borland	Data Base File	Ja	Für dBase-Datenbanken. Weit verbreiten, vor allem im MS-DOS Bereich. Als Seriendruckdatenquelle geeignet.
Dib	Grafik	Hersteller unabhängig	Device Independet Bitmap	Ja	Ist dem BMP-Format sehr ähnlich.
Doc	Textprozessor	Microsoft	Document	Ja	Diese Dateierweiterung wird bzw. wurde auch noch von anderen verwendet.
Dot	Textprozessor	Microsoft	Document Template	Ja	Dokumentvorlage für MS Word.
Drw	Grafik	Micrografx	Draw	Ja	---
Dxf	Grafik	Autodesk	Drawing Exchange Format	Ja	Wird für AutoCAD verwendet (Computer Aided Design). Soll in Zukunft abgelöst werden.
Emf	Grafik	Microsoft	Windows Enhanced Metafile	Ja	Variante von Windows Metafile (siehe WMF) aber moderner. Erfordert dann auch eine 32-Bit Architektur.

Datei-Endung	Genre	Hersteller	Steht für	Filter in MS Word	Bemerkungen
Eps	Grafik	Hersteller unabhängig	Encapsulated PostScript	Ja	---
Fpx	Grafik	Hersteller-konsortium	FlashPix	Ja	Neues Grafikformat, gemeinsam definiert durch Apple, Canon, Hewlett Packard, IBM, Intel, Kodak, Live Picture und Microsoft. Noch nicht sehr verbreitet.
Gif	Grafik	CompuServe	Graphics Interchange Format	Ja	Komprimierbares Grafikformal (vektororientiert). Heute immer noch weit verbreitet. Von CompuServe freigegeben.
htm html	Textgestal-tung	Hersteller unabhängig	Hypertext Markup Lamguage	Ja	Textgestaltung basierend auf "tagging* statt Format. Weit ver-breitet im Internet. Transparent und sehr robust. Das Resultat des Imports nach MS Word ist gut. Der Export aus MS Word nach HTML ist zwar unterstützt aber im Resultat eine Katastrophe (chaotische, intransparenter und aufgeblasener HTML-Code).
Ico	Sinnbilder	Microsoft	Icon	Nein	Wird für Symbole (Sinnbilder) auf der PC-Oberfläche verwendet.
jpg jpeg	Grafik	Hersteller unabhängig	Joint Photografic Expert Group	Ja	Komprimierbares Grafikformat. Heute weit verbreitet.
Lwp	Textpro-zessor	Lotus heute Tochter von IBM	Lotus WordPro	Nein	Siehe dazu den entsprechenden Abschnitt in diesem Skript.
Pdf	Publikatio-nen	Adobe	Portable Document Format	Nein	Lässt sich (bewusst) nicht importieren. Das Erstellen verlangt den Adobe Acrobat oder vergleichbare Software. Siehe dazu die entsprechenden beiden Kapitel in diesem Skript.
Png	Grafik	World Wide Web Konsorti-um	Portable Network Graphic For-mat	Ja	Sollte sowohl GIF wie JPEG ablösen, verbreitet sich aber nur sehr langsam.
Poa	Präsentati-on	Microsoft	PowerPoint Add-In	---	Diese Dateierweiterung wird für PowerPoint Add-Ins verwendet.
Pot	Präsentati-on	Microsoft	PowerPoint Template	---	Diese Dateierweiterung wird für PowerPoint Vorlagen verwendet.
Ppt	Präsentati-on	Microsoft	PowerPoint	---	Diese Dateierweiterung wird für PowerPoint Präsentationen verwendet.
Rtf	Textaus-tausch	Microsoft	Rich Text Format	Ja	Von Microsoft für den Ex- und Import von Texten entwickeltes Textformat. Nicht verwechseln mit RFT-DCA von IBM! Word kann diese Dateien auch konvertieren. Dazu ist aber ein externer Filter zu verwenden, welcher im Converter-Pack von Microsoft enthalten ist.
Sam	Textpro-zessor	Ami	Ami Pro	Ja	Unterstützt werden die Versionen Ami Pro 2.0 und Ami Pro 3.0. Dokumente, welche mit dem Nachfolgeprodukt Lotus WordPro geschrieben wurden werden allerdings nicht unterstützt. Siehe dazu die Dateierweiterung lwp.
Swd	Textpro-zessor	Sun	StarWriter	Nein	Bestandteil von StarOffice™. Ursprünglich von Star Division entwickelt. Siehe dazu den entsprechenden Abschnitt in diesem Skript.
tif tiff	Grafik	Hersteller unabhängig	Tagged Image File Format	Ja	---

Datei-Endung	Genre	Hersteller	Steht für	Filter in MS Word	Bemerkungen
Txt	Text	Hersteller unabhängig	Text	Ja	Man muss unterscheiden zwischen Textdateien, welche in ASCII codiert sind und andere, welche in ANSI codiert sind. Die ersteren Dateien werden im Microsoft-Umfeld unter MS-DOS erstellt, die letzteren unter Windows. Microsoft hat zeitweise auch den sogenannten "Erweiterten ASCII-Code" propagiert und verwendet, dabei handelte es sich aber um einen Alleingang, welcher Microsoft mit der Einführung von Windows aufgegeben hat. Der ursprünglich ASCII-Code ist noch ein 7-Bit Zeichencode (max. 128 Zeichen), die anderen beiden sind 8-Bit Code (256 Zeichen). Lesen Sie auch die Kapitel über MS-DOS Dateien in diesem Skript.
Wbk	Textpro-zessor	Microsoft	Word Backup	---	Sicherungsdatei für MS Word-Dokumente.
Wdb	Datenbank	Microsoft	Works Data Base	Nein	Kann nicht als Datenquelle für den Seriendruck in MS Word herhalten. Sie müssen die Datei in Works in einem MS Word kompatiblen Format abspeichern.
Wiz	Textpro-zessor	Microsoft	Word Wizard	---	Assistenten für MS Word
wk1-4	Tabellen-kalkulation	Lotus im Besit-ze von IBM	Work Sheet	Ja	---
Wks	Textpro-zessor	Microsoft	Works	Ja/Nein	Works 3.0 und Works 95 sind unterstützt. Works2000/2001 ist (noch) nicht unterstützt. Speichern Sie diese Dateien unter Wors2000(2001 unter einem Word-kompatiblen Format ab, um sie in MS Word zu öffnen.
Wps	Textpro-zessor	Microsoft	Works	Ja/Nein	Siehe wks.
Wll	Textpro-zessor	Microsoft	Word Add-In	---	Add-Ins für MS Word
Wmf	Grafik	Microsoft	Windows Metafile Format	Ja	Sowohl pixel- wie auch vektororientiert. Zum Datenaustausch zwischen Anwendungen konzipiert.
wp4-6	Textpro-zessor	Corel	WordPerfect	Ja/Nein	Das Produkt wechselten in den Vergangenheit oft den Besitzer. (WordPerfect ---> Novell, 1994 ---> Corel, 1996) MS Word kann folgende Versionen lesen: WordPerfect 4.1, 4.2, 5.x und 6.x. Höhere Versionen werden wohl auch in Zukunft aus Konkurrenz-denken von Microsoft nicht unterstützt.
Wir	Textpro-zessor	Microsoft	Write	Ja	Wird von WordPad benutzt. Format aus den ersten Stunden der elektronischen Textverarbei-tung unter MS-DOS und später Windows.
Ws1-7	Textpro-zessor	WORDSTAR	WordStar	Ja	MS Word kann folgende Versionen einlesen: WordStar 1.0 - 2.0 für Windows und WordStar 3.3 - 7.0.
Xla	Tabellenka lkulation	Microsoft	Excel Add-In	---	Add-Ins für MS Excel
Xls	Tabellenka lkulation	Microsoft	Excel Work Sheet	Ja	Diese Dateierweiterung wird für Excel Arbeitsmappen eingesetzt. Kann direkt in ein Word-Dokument importiert werden. Auch als Datenquelle für den Seriendruck in MS Word geeignet. Alternativ können Daten aus Excel statt über den Excel Konver-ter auch mit DDE oder ODBC importiert werden.
Xlt	Tabellenka lkulation	Microsoft	Excel Template	---	Vorlagen für MS Excel

VBA-technischer Umgang mit externen Word-Konvertern

▲ Zurück zum Anfang

Festellen welche Konverter auf dem System vorhanden sind

Word kennt jeden anwendbaren Konverter unter einer Nummer. Diese Nummer ist allerdings installationsabhängig. Auf Ihrem Desktop-PC lautet diese Zahl für das Format "MS-DOS-Text" vielleicht "6" während auf Ihrem Laptop der gleiche Konverter unter der Nummer "4" erscheint. Aus diesem Grund müssen Sie einen indirekten Weg beschreiten.

Der folgende Code ermittelt alle auf dem System installierten Konverter, welche sich zum Importieren einer Datei eigenen und stellt diese in einer Tabelle dar. Diese Tabelle verrät Ihnen drei Dinge, nämlich welche Konverter auf einem bestimmten System zur Verfügung stehen und unter welcher ID diese auf diesem spezifischen System anzusprechen sind. Zudem wird für jeden der Konverter noch die dazugehörige Dateierweiterung ausgewiesen.

Kopieren Sie den nachfolgenden Code in der VBA-Umgebung in ein neues Modul und führen Sie ihn aus.

```
Sub KonverterAuflisten()
  Set rDoc = Documents.Add
  Dim oConv As FileConverter
  Selection.TypeText Text:="Name" & vbTab & "Kennung" & vbTab & _
     "Erweiterung" & vbCrLf
  For Each oConv In Application.FileConverters
    If oConv.CanOpen = True Then
      Selection.TypeText Text:=oConv.FormatName & vbTab & _
         oConv.OpenFormat & vbTab & oConv.Extensions & vbCrLf
    End If
  Next
  With Selection
    .WholeStory
    .ConvertToTable Separator:=wdSeparateByTabs, _
        Format:=wdTableFormatProfessional, AutoFit:=True
    .HomeKey Unit:=wdStory
  End With
End Sub
```

(Dieser Code generiert eine Tabelle, welche Ihnen bezüglich der auf Ihrem System installierten Konverter einen Überblick verschafft. Die Kennung der einzelnen Konverter ist auf jedem Rechner anders.)

Das Ergebnis präsentiert sich in etwa wie folgt:

Format Name	Kennung	Erweiterung
Text mit Layout	15	ans
MS-DOS-Text mit Layout	10	asc
Text aus beliebiger Datei wiederherstellen..	4	.*
Lotus 1-2-3	8	wk1 wk3 wk4
Microsoft Excel-Arbeitsblatt	9	xls xlw
Word 4.0 - 5.1 für Macintosh	18	mcw
WordPerfect 6.x	22	wpd doc
WordPerfect 5.x	21	doc
Persönliches Adressbuch	12	pab
Outlook-Adressbuch	11	olk
Windows Write	17	wri
Word 6.0/95 für Windows/Macintosh	20	doc

! Beachten Sie, dass MS Word sogenannte anwendungsintegrierte Konverter kennt (z.B. für RTF). Für diese Dateien ist kein zusätzlicher, also externer Konverter nötig. Somit ist dafür auch keine Konverterklasse in der Registrierdatenbank eingetragen was wiederum zur Folge hat, dass diese Filter nicht in der oben gezeigten Liste erscheinen.

☹ Die oben abgedruckte Liste wurde von mir geläutert. Microsoft bringt es nämlich fertig, ein ziemliches Chaos anzurichten. So kann z.B. der Filter mit der Bezeichnung *MS-DOS-Text mit Layout* mehrmals vorkommen, also z.B. auch noch unter *MS-DOS-Text with Layout.* Ausgehend von den Dateierweiterungen statt vom Klassennamen, ist die Sache auch nicht besser,. kann doch der selbe Textkonverter einerseits unter *asc* sowie auch unter *ASC* registriert sein. Nach den Unterlagen, welche mir von Microsoft vorliegen, ist *MS-DOS-Text with Layout* tatsächlich ein gültiger Klassenname. Wieso die Leute von Microsoft auf die Idee kommen solch unhandliche, inkonsistente und sprachenabhängige Klassennamen zu vergeben, weiss ich leider auch nicht. Man denkt dort vermutlich: *It's good enough*. Auf mich wirkt das eher schlampig bzw. unprofessionell.

Die Konverterkennung für ein bestimmtes Dateiformat finden

Gehen wir davon aus, Sie möchten eine Datei als *MS-DOS-Text mit Layout* öffnen, dann sollte Sie eine Mimik zur Hand haben, welche die Kennung, welche ja auf jedem Rechner unterschiedlich ist, für dieses Format finden kann. Je nachdem, können Sie entweder vom Formatnamen ausgehen, wie dies die folgende Routine zeigt, oder Sie können auch über die Dateierweiterung gehen. Diese Mimik sieht dann wie folgt aus:

```
Sub KonverterIDErmitteln()
  Konverter = "MS-DOS-Text mit Layout"
  Dim oConv As FileConverter
  For Each oConv In Application.FileConverters
    If oConv.CanOpen = True And oConv.FormatName = Konverter Then
      Conv = oConv.OpenFormat
      Exit For
    End If
  Next
  If Conv = "" Then
    MsgBox "Der Konverter " & Konverter & " _
       ist nicht auf dem Rechner installiert oder eignet sich nicht zum Öffnen."
  Else
    MsgBox "Die ID lautet: " & Conv
  End If
End Sub
```

(Die Konverter-ID für das Format "MS-DOS-Text mit Layout" wird dynamisch ermittelt.)

Alles unter einen Hut bringen

Nun bringen wir unsere Erkenntnisse zusammen. Der folgende Code ermittelt zuerst die I**D**, welche wir für das korrekte Öffnen brauchen und öffnet anschliessend die Datei.

```
Sub DosDateiOeffnen()
  CnvID = KonverterIDErmitteln("MS-DOS-Text mit Layout")
  Documents.Open FileName:="C:\AutoExec.bat", Format:=CnvID
End Sub

Private Function KonverterIDErmitteln(tFormat)
  For Each oConv In Application.FileConverters
    If oConv.CanOpen = True And oConv.FormatName = tFormat Then
      Conv = oConv.OpenFormat
      Exit For
    End If
  Next
  If Conv = "" Then
    MsgBox "Der Konverter " & Konverter & " _
       ist nicht auf dem Rechner installiert oder eignet sich nicht zum Öffnen."
    End
  Else
    KonverterIDErmitteln = Conv
  End If
End Function
```

(Öffnet eine *DOS-Datei mit Layout* im richtigen Format. Der Code ermittelt die Konverter-ID dynamisch.)

! Es wird Ihnen nicht gelingen, eine MS-DOS-Datei, so zu öffnen, dass sowohl Umlaute, wie auch Zeilenumbrüche etc. richtig interpretiert werden. Schauen Sie sich dazu das Kapitel «*MS DOS-Datei mit Word-VBA öffnen*» in diesem Skript an.

Dokument in einem bestimmten Format speichern

Hier ist noch das Gegenstück dazu. Der folgende Code ermittelt zuerst die ID, welche wir für das korrekte Speichern brauchen und speichert anschliessend die Datei.

```
Sub DosDateiSpeichern()
  CnvID = KonverterIDErmitteln("MS-DOS-Text mit Layout")
  ActiveDocument.SaveAs FileName:="C:\DosText.asc", FileFormat:=CnvID
End Sub

Private Function KonverterIDErmitteln(tFormat)
  For Each oConv In Application.FileConverters
    If oConv.CanSave = True And oConv.FormatName = tFormat Then
      Conv = oConv.SaveFormat
      Exit For
    End If
  Next
  If Conv = "" Then
    MsgBox "Der Konverter " & Konverter & " _
       ist nicht auf dem Rechner installiert oder eignet sich nicht zum Speichern."
    End
  Else
    KonverterIDErmitteln = Conv
  End If
End Function
```

(Speichert eine *DOS-Datei mit Layout* im richtigen Format. Der Code ermittelt die Konverter-ID dynamisch.)

Links zu diesem Thema

▲ Zurück zum Anfang

Microsoft

Deutsch

► WD2000: Konvertierung zwischen Lotus Word Pro 96 und Word
► Konvertieren von Adobe Acrobat Reader-Dateien (*.pdf-Dateien)

Englisch

► WD97: Text Converters Installed in a Typical Setup
► WD2000: Text Converters Installed During an "Install Now" Setup
► WD97: Supported Non-Word File Formats
► WD97: Graphics Filters Installed During Typical Setup
► WD2000: Graphics Filters Installed in an "Install Now" Setup
► WD: Additional Text Converters and Image Filters Available in Microsoft Office Converter Pack
► WD97: How to Use Other Converters with the File Format Property
► WD2000: How to Use Other Converters with the File Format Property
► Microsoft Word2000 Help - Class names of file converters
► WD: History of WinWord File Converters Categorized by Release
► WD97: Difference Between RTF and RFT File Formats

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Konvert.htm

René Probst, Oktober 1999

Änderungen

- 7. Februar 2001 - Batch Konvertierung
- 7. Februar 2001 - Neu strukturiert
- 1. März 2001 - Zahlreiche Erweiterungen

Ist der Passwortschutz für Word 97 und Word2000 Dokumente sicher?

◄ Zurück zum Hauptindex

Frage

> Wenn ich mein Word 97 oder Word2000 Dokument mit einem Passwort schütze ist es dann sicher vor fremden Zugriff oder muss ich damit rechnen, dass jemand es mit einem (simplen) Trick doch öffnen kann?

Antwort

Wenn Sie bei der Vergabe des Passwortes auf drei oder vier Dinge achten, dann ist das Dokument sehr sicher und dies sogar vor Ihrem eigenen Zugriff, sollten Sie das Passwort einmal vergessen bzw. verlegen! Der Schutz basiert auf einem bekannten Algorithmus, welcher nach Meinung aller Fachleute nur durch das Versuch-und-Irrtum Verfahren zu knacken ist. Da aber Word sowohl Buchstaben, Zahlen und die meisten Sonderzeichen zulässt und dazu Gross- und Kleinschreibung unterschiedlich interpretiert, wäre ab einer Passwortlänge von sieben bis acht Stellen mehr Rechenleistung erforderlich, als dies ein PC binnen Monaten oder Jahre hergeben kann.

Am idealsten wäre es, Sie würden das Passwort gar nicht selbst ausknobeln, sondern diese Arbeit einem Programm überlassen, welches wirklich nach dem Zufallsprinzip vorgeht. Ein solches kleines Programm ist unten eingeblendet.

Sie kommen aber auch ohne Programm zum Ziel. Achten Sie einfach auf die folgende Punkte:

- Vergeben Sie mindestens 7 Zeichen (das Maximum liegt übrigens bei 15 Zeichen).
- Bauen Sie an zufälliger Stelle das eine oder andere Sonderzeichen ein.
- Vermeiden Sie Reihen, insbesondere leicht durchschaubare, wie z. B. August08.
- Vermeiden Sie triviale Passwörter, wie z. B. Supervisor, Admin oder Chef.
- Vermeiden Sie assoziative Passwörter, wie z. B. Susi, Flipper, Diana, Audi, Schumi, etc.
- Vermeiden Sie alles, was ein sinnvolles und korrektes Wort ergibt.

Anmerkungen

- Der Passwortschutz von früheren Word-Versionen gilt als **nicht** sicher.
- Falls Sie Ihr eigenes Passwort vergessen haben, versuchen Sie sich daran zu erinnern.

Gegebenheiten in Frankreich

Der Algorithmus des Passwortschutzes wie er in der angelsächsischen und in der deutschen Office-Version implementiert ist, wurde in Frankreich auf Druck der Regierung (wahrscheinlich steckt der Geheimdienstes dahinter) verboten. Der Passwortschutz, welcher in Frankreich zur Anwendung kommt, ist primitiver und somit unsicher. Als Konsequenz davon kann ein Office-Dokument, welches z. B. in der deutschen Programmversion verschlüsselt wurde, in Frankreich nicht geöffnet werden, und zwar auch dann nicht, wenn Sie das Passwort wissen bzw. kommunizieren. Der einzige Ausweg hier ist es, das Dokument unverschlüsselt zu übergeben. Der Empfänger kann es dann nach den in Frankreich geltenden Regeln verschlüsseln.

Passwortknacker

Ich wiederhole es nochmals:

Es gibt keinen verlässlichen Weg, ein mit Passwort geschütztes Word 97-Dokument zu öffnen, es sei denn, man kenne dass richtige Passwort.

Im World Wide Web werden Programme angeboten, welche das Passwort ***angeblich*** knacken können. Mit der Demo-Version können in der Regel Passwörter bis zu einer Länge von maximal drei oder vier Zeichen geknackt werden. Das klappt dann auch immer, da die Kombinationsmöglichkeiten in dieser Konstellation noch ziemlich beschränkt sind. Ich habe Verständnis dafür, dass eine Demo-Version mit Einschränkungen verbunden ist, würden sonst doch nur noch wenige Leute die Vollversion käuflich erwerben.

Weniger Verständnis habe ich, für Gaukler und Geldmacherei. Überzeugen würde mich eine Demo-Version, welche zwar nur die ersten drei Zeichen des Passwortes sichtbar macht, das Passwort aber unabhängig von der Länge und von der Komplexität der Zeichenfolge, intern, in Tat und Wahrheit geknackt hat.

Genau dies ist aber nicht der Fall. Die Anbieter brüsten sich, wie viele Passwörter das Programm pro Sekunde durchprobiert. Es fehlt aber jeglicher Hinweis, welche astronomische Zahl von Versuchen nötig wären um statistisch gesehen ein Passwort mit der Wahrscheinlichkeit 1:1 zu knacken. Die Millionen oder Milliarden von Versuchen, welcher ein Pentium-Prozessor zu Stande kriegt, sind im Vergleich zu 200 hoch 15, als würde man die Sonne mit Erbsen bombardieren.

Im Hinblick der Fairness, muss hier noch erwähnt werden, dass diese Programme in der Regel auch in Kombination mit Wörterbüchern arbeiten können. Dabei wird aber stillschweigend spekuliert, dass Menschen Zeichenfolgen für Passwörter nicht wirklich zufällig vergeben, sondern assoziativ.

VBA-Code zur Generierung eines Zufallcodes

```
Sub ZufallsZeichenfolge()
  Länge = 10
  Dim ZFolge As String
  Rnd
  For i = 1 To Länge
    X = Int((255 * Rnd))
    If X > 40 Then
      ZFolge = ZFolge & Chr(X)
    Else
      i = i - 1
    End If
  Next i
  MsgBox ZFolge
End Sub
```

(Generierung eines zufälligen Passwortes)

Anmerkungen für Besserwisser

- Bei der Tatsache, dass sich das Passwort weder knacken noch umgehen lässt handelt es sich nicht um einen BUG, es ist von den Erfindern so gewollt.
- Die Informationen in diesem Skript basieren auf ziemlich gründlichen Recherchen.
- Ich bin **nicht interessiert** an Erwiderungen nach dem Muster *es lässt sich doch alles knacken*. Diese Aussage ist längsten widerlegt und zwar sowohl im technischen, wie auch im juristischen Sinne. So basiert z.B. auch PGP (Pretty Good Privacy) auf einem Algorithmus, welcher nachweislich sicher ist.
- Ich bin weiter nicht interessiert an Vergleichen mit Super-Computern, insbesondere da die Leistung eines modernen Pentiumrechners nicht zu verachten ist. Grossrechner von IBM und Amdhal können zwar riesige Datenmengen bewältigen, sind aber nicht auf komplexe Berechnungen ausgelegt. Der Bau des Supercomputers *Cray* wurde vor Jahren eingestellt, weil es heute kostengünstiger kommt z.B. 64 Pentium-Rechner parallel zu betreiben, um so eine hohe Rechenleistung zu erzielen.
- Man kann natürlich immer philosophieren, ob sämtliche Fachleute versagt haben und doch noch einer hinter den Trick kommen wird, wie die Sache zu umgehen ist. Ich glaube jedoch, dass die meisten Leute eine konkrete, kompetente und sachdienliche Antwort suchen und nicht solche Binsenwahrheiten.
- Zusammengefasst: hört auf mich mit Mails, welche Allgemeinplätze enthalten und mir Naivität vorwerfen zu bombardieren. Recherchiert doch zuerst selbst einmal. Es reicht wahrscheinlich nicht, sich in einem TV-Magazin eine halbstündige Sendung über Hacker anzugucken.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Passwort.htm

René Probst, Oktober 1999

Änderungen

- 3. Dezember 1999 - Situation in Frankreich
- 3. Dezember 1999 - Passwortknacker
- 24. April 2000 - Besserwisserei
- 14. November 2000 - Auch nach über einem Jahr gibt es keine Hinweise darauf, dass in Word2000 etwas verschlimmbessert wurde

Wie kann ich einem Word-Dokument oder in einem Word-Dokument eine fortlaufende Nummer vergeben?

◄ Zurück zum Hauptindex

Problemstellung

> 1. Ich möchte ein Dokument, unter Vergabe einer im Dokument sichtbaren Laufnummer, mehrmals drucken.
>
> 2. Ich möchte für Dokumente, welche auf der ein und der selben Dokumentenvorlage beruhen, an geeigneter Stelle eine fortlaufende Nummer hinzufügen und diese auch unter dieser Nummer abspeichern.

Diese Seite

▼ Lösungsweg
▼ Neue Dokumenteigenschaft erstellen
▼ Das Dokument x-Mal zu drucken unter Vergabe einer sichtbaren Laufnummer im Dokument
▼ Laufnummer speziell formatieren
▼ Vergabe einer Laufnummer bei der Generierung eines Dokuments (Variante Registrierdatenbank)
▼ Vergabe einer Laufnummer bei der Generierung eines Dokuments (Variante INI-Datei)
▼ Zurücksetzen der Laufnummer
▼ Links zu diesem Thema

Lösungsweg

▲ Zurück zum Anfang

Es gibt eine ganze Reihe von guten Lösungsansätzen für diese Problemstellung. Dabei kommt es darauf an, ob der augenblickliche Stand der Nummer auch über ein Word- bzw. Windows-Restart hinaus, erhalten bleiben muss. Es macht auch ein Unterschied, ob ein und das selbe Dokument x-Mal mit einer aufsteigenden Nummer gedruckt werden soll, oder ob die Laufnummer bei der Erstellung eines neuen Dokumentes vergeben werden soll und mit jedem einzelnen Dokument abgespeichert werden sollen. Nachfolgend werden zwei Lösungen erörtert.

Beide Vorschläge arbeiten mit einer sogenannten benutzerdefinierten **Dokumenteigenschaft,** welche Sie dem Dokument bzw. der Dokumentenvorlage, auf welcher die Dokumente basieren, an geeigneter Stelle, z. B. im normalen Textfluss oder in einem Textfeld referenzieren können. Der Name der Dokumenteigenschaft soll in unseren Beispielen **Laufnummer** heissen. Bitte beachten Sie, dass dieser Name mit den Definitionen im Code übereinstimmen muss. Dies bedeutet, dass wenn Ihnen dieser Name nicht behagt, dann müssen Sie auch den VBA-Code anpassen. Der Name der Eigenschaft hat übrigens keinerlei Einfluss auf das Erscheinungsbild.

Der Grund, wieso ich mich für eine Dokumenteigenschaft und nicht für eine Dokumentvariable (welche fast den selben Zweck erfüllen) entschieden habe, ist dieser, dass Word 97 beim Einfügen des Feldes *DokVariable* in Kopf-/und Fusszeilen oder in Textfeldern reproduzierbar abschmiert (*Great, Microsoft!*). Dieses gravierende Fehlverhalten wird nicht mit den Service Releases von Office 97 gelöst, sondern erst mit Word2000.

Neue Dokumenteigenschaft erstellen

▲ Zurück zum Anfang

Hier nun die nötigen Schritte, um die Dokumenteigenschaft mit den Namen **Laufnummer** Ihrem Dokument oder Ihrer Dokumentenvorlage hinzuzufügen.

1. **Öffnen** Sie die Dokumentvorlage oder das Dokument (***Datei -Öffnen***)
2. Setzen Sie im Dokument oder in der Dokumentvorlage die Einfügemarke an den Ort, an der die Laufnummer erscheinen soll. Diese Position darf sich auch innerhalb eines Textfeldes oder in der Kopf-/Fusszeile befinden.
3. Vom Word-Hauptmenü wählen Sie nun den Dialog ***Einfügen - Feld...***.
4. In der linken Liste mit der Bezeichnung *Kategorien:* markieren sie den Eintrag **Dokumentinformationen**.
5. In der rechten Liste mit der Bezeichnung *Feldnamen:* markieren Sie den Eintrag **DokEigenschaft** falls Sie **Word 97** benutzen oder aber **DocProperty** falls Sie **Word2000** benutzen.
6. Als nächstes sollten Sie das Textfeld in diesem Dialog ergänzen. Setzen Sie also die Einfügemarke hinter den Feldnamen DokEigenschaft (bzw. DocProperty), achten Sie darauf, dass eine Leerstelle vorhanden ist und tippen sie **"Laufnummer"** und zwar **inklusive Gänsefüsschen**. Der ganze Ausdruck heisst nun also: **DokEigenschaft "Laufnummer"** bzw. **DocProperty "Laufnummer"**.
7. Schliessen Sie nun den Dialog mit ***Ok*** ab. Word wird einen Fehler monieren, da es diese Dokumenteneigenschaft im Moment gar noch nicht gibt, sondern nur die Referenz darauf. Dies soll uns aber nicht stören; die Eigenschaft wird später mittels VBA-Code erstellt.
8. Sichern Sie nun das Dokument oder die Dokumentenvorlage.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

Das Dokument x-Mal zu drucken unter Vergabe einer sichtbaren Laufnummer im Dokument

▲ Zurück zum Anfang

Erstellen Sie eine Dokumentvariable, wie dies vorgängig Schritt für Schritt besprochen wurde. Kopieren Sie dann den folgenden Code in die Dokumentvorlage.

```
Private Const Titel = "Dokument drucken"

Sub HochzählenUndDrucken()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Projekt = oDoc.AttachedTemplate.VBProject.Name
  If oDoc.Path = "" Then
    aSec = oDoc.Name
  Else
    aSec = Left(oDoc.Name, InStr(oDoc.Name, ".") - 1)
  End If
  Num = GetSetting(AppName:=Projekt, Section:=aSec, _
     Key:="Laufnummer", Default:="0")
  Num = Num + 1
  Flag = False
  While Flag = False
    Antwort = InputBox("Mit welcher Laufnummer möchten Sie beginnen?", Titel, Num)
    If Antwort = "" Then End
    If IsNumeric(Antwort) Then
      Flag = True
    Else
      MsgBox "Der eingegebene Wert ist keine gültige Zahl.", vbCritical, Titel
    End If
  Wend
  Num = Int(Antwort)
  Flag = False
  While Flag = False
    Antwort = InputBox("Wieviele Kopien möchten Sie drucken?", Titel, 1)
    If Antwort = "" Then End
    If IsNumeric(Antwort) Then
      Flag = True
    Else
      MsgBox "Der eingegebene Wert ist keine gültige Zahl.", vbCritical, Titel
    End If
  Wend
  On Error Resume Next
  oDoc.CustomDocumentProperties.Add Name:="Laufnummer", Value:="", _
     LinkToContent:=False, Type:=msoPropertyTypeString
  On Error GoTo 0
  For i = 1 To Int(Antwort)
    oDoc.CustomDocumentProperties("Laufnummer").Value = Num
    For Each Teil In oDoc.StoryRanges
      Teil.Fields.Update
      While Not (Teil.NextStoryRange Is Nothing)
        Set Teil = Teil.NextStoryRange
        Teil.Fields.Update
      Wend
    Next
    oDoc.PrintOut Background:=False
    Num = Num + 1
  Next i
  Num = Num - 1
  SaveSetting AppName:=Projekt, Section:=aSec, _
     Key:="Laufnummer", Setting:=Num
  End Sub
```

(Druckt ein Dokument unter Vergabe einer im Dokument sichtbaren Laufnummer x-Mal. Der Stand der Laufnummer wird in der Registrierdatenbank aufbewahrt.)

Bemerkungen zum Code

- Das Makro ist interaktiv. Sie werden gefragt, mit welcher Laufnummer die Chose beginnen soll und wie viele aufsteigende Kopien gedruckt werden sollen.
- Kopieren Sie den Code in der VBA-Umgebung in ein neues Modul des fraglichen Dokumentes.
- Fügen Sie dem Dokument eine Dokumenteigenschaft mit dem Namen **Laufnummer** hinzu, wie dies Eingangs beschrieben wurde.
- Sie möchten die Zahl speziell formatiert haben, z.B. wünschen Sie führende Nullen oder sogar Römische Zahlen, dann lesen sie das folgende Kapitel.

Laufnummer speziell formatieren

▲ Zurück zum Anfang

Wenn Sie diese Laufnummer speziell formatieren wollen, dann bietet sich ein sogenannter Feldschalter für das Feld *DokEigenschaft* bzw. *DocProperty* an.

Als erstes sollten Sie sich mit der Tastenkombination [Alt+F9] den Feldcode des Feldes anzeigen lassen. Ist dies geschehen, dann können Sie den Feldschalter dem Feldcode zu fügen und zwar wie folgt:

Vierstellige Zahl links mit Nullen auffüllen

Ergänzen Sie den Feldcode wie folgt:

```
{ DokEigenschaft "Laufnummer" \# "0000" }
```
Word 97

```
{ DOCPROPERTY "Laufnummer" \# "0000" }
```
Word2000

Das Resultat sieht dann als Beispiel wie folgt aus: **0013**

Römische Zahlen

Sollten Sie den Auftrag direkt von Julius Caesar erhalten haben, dann ergänzen Sie den Feldcode wie folgt:

```
{ DokEigenschaft "Laufnummer" \* Roman }
```
Word 97

```
{ DOCPROPERTY "Laufnummer" \* Roman }
```
Word2000

Das Resultat sieht dann als Beispiel wie folgt aus: **XIII**

☛ Sie können den Feldschalter «*Roman*» auch mit einem kleine **«r»** schreiben, dann wird eine klein geschriebene römische Ziffer angezeigt.

VBA-Code zur Vergabe einer Laufnummer bei der Generierung eines Dokuments, welches auf eine bestimmte Dokumentvorlage beruht.

▲ Zurück zum Anfang

Im folgenden werden zwei Lösungen vorgestellt, welche bei der Generierung eines Dokumentes, welches auf einer bestimmten Dokumentvorlage beruht, eine fortlaufende Zahl (Kollerator) ermitteln.

Die erste Lösung setzt dabei auf einen **privaten Schlüssel in der Registrierdatenbank.** Sollte sich dieser Weg als untauglich erweise, vielleicht weil diesem Schlüssel in Ihrer vernetzten Umgebung keine zentrale Bedeutung zukommt, dann können Sie auf die zweite Lösung ausweichen, welche den aktuellen Stand dieser Laufnummer in einer Art I**NI-Datei** nachführt. Die Meinung ist, dass Sie diese Datei so anlegen, dass ihr eine zentrale Bedeutung zukommt.

Variante in Zusammenarbeit mit der Registrierdatenbank

```
Private Const Präfix = "Rechnung"

Sub AutoNew()
  Dim oDoc As Document, oV As Template
  Set oDoc = ActiveDocument
  Set oV = oDoc.AttachedTemplate
  Projekt = oV.VBProject.Name
  aSec = aSec & Left(oV.Name, InStr(oV.Name, ".") - 1)
  Num = GetSetting(AppName:=Projekt, Section:=aSec, _
     Key:="Laufnummer", Default:="0")
  Num = Val(Num) + 1
  SaveSetting AppName:=Projekt, Section:=aSec, _
     Key:="Laufnummer", Setting:=Num
  On Error Resume Next
  oDoc.CustomDocumentProperties.Add Name:="Laufnummer", _
     LinkToContent:=False, Value:=Num, Type:=msoPropertyTypeString
  On Error GoTo 0
  For Each Teil In oDoc.StoryRanges
    Teil.Fields.Update
    While Not (Teil.NextStoryRange Is Nothing)
      Set Teil = Teil.NextStoryRange
      Teil.Fields.Update
    Wend
  Next
  With Dialogs(wdDialogFileSummaryInfo)
    .Title = Präfix & Num
    .Execute
  End With
  oDoc.Windows(1).Caption = Präfix & Num
End Sub
```

(Dieser Code ist Bestandteil einer Dokumentvorlage. Beim Generieren eines neuen Dokumentes wird eine Laufnummer vergeben. Wird das Dokument nun vom Benutzer gespeichert, wird ein Dateiname vorgeschlagen, welcher sich aus einem Präfix und einer fortlaufenden Nummer zusammen setzt. Die Laufnummer wird in einem Schlüssel der Registrierdatenbank gespeichert.)

Bemerkungen zum Code

- Kopieren Sie den Code in der VBA-Umgebung in ein neues Modul der fraglichen Dokumentvorlage.
- Wenn Sie die Laufnummer im Dokument anzeigen wollen, fügen Sie der Dokumentvorlage an geeigneter Stelle eine Dokumenteigenschaft mit dem Namen **Laufnummer** hinzu, wie das Eingangs in diesem Skript beschrieben wurde.
- Sie können den **Präfix** für den Namen unter dem das Dokument gespeichert wird, jederzeit im Kopf des Codes anpassen.
- Sie können die Laufnummer auf Wunsch nachformatieren. Falls Sie eine vierstellige Zahl benötigen, welche links mit Nullen aufgefüllt ist, dann erreichen Sie dies mittels VBA wie folgt: `Num = Format(Num, "0000").`
- Damit der Prozess bei jedem neuen Dokument aufgerufen wird, ist der Name des Prozesses, nämlich **AutoNew**, zwingend.
- Der Code ist so gestaltet, dass es potentiell für jede Dokumentvorlage einen Zähler gibt. Sie können diese Mimik allerdings überlisten, falls Sie in verschiedenen Verzeichnissen eine Dokumentvorlage mit dem selben Namen haben und diese Vorlagen entsprechend ausrüsten. Diesen Spezialfall könnten Sie allenfalls auch abdecken, wenn Sie dem Projekt einen eindeutigen Namen vergeben. Wie man einem Projekt einem eigenen Namen vergibt, erfahren Sie gleich anschliessend.
- Die Laufnummer wird übrigens in der Registrierdatenbank unter einem Schlüssel, welcher folgendes Muster hat abgespeichert:
  `HKEY_CURRENT_USER\Software\VB and VBA Program Settings\`***`Projektname`***`\`***`DOTName`***

Dem Projekt einen eigenen Namen vergeben

Es ist zwar in den allermeisten Fällen nicht zwingend aber eine gute Idee, wenn Sie dem Projekt einen eigenen und eindeutigen Namen vergeben. Die Eindeutigkeit des Namens müssen übrigens Sie sicherstellen (will heissen Sie sollten nicht zweimal den selben Projektname vergeben).

Jede Dokumentvorlage und jedes Dokument, welches Makroelemente beinhaltet, kann als **Projekt** angesehen werden und die vorhandenen Elemente können ohne Umweg über den Pfad-/Dateinamen auch über einen **Projektnamen** angesprochen werden. In der Regel handelt es sich um eine 1:1 Relation (will heissen eine Dokumentvorlage = ein Projekt). Standardmässig vergibt Word aber für jedes Projekt den selben Namen, womit das Ansprechen über den Projektnamen nicht möglich bzw. nicht sinnvoll ist. Um dem abzuhelfen gehen Sie wie folgt vor:

1. **Öffnen** Sie die fragliche Dokumentvorlage mit ***Datei - Öffnen***.
2. Wechseln Sie mit der Tastenkombination **[Alt+F11]** in die VBA-Umgebung.
3. Lassen Sie sich mit der Tastenkombination [**Strg+R**] den Projekt-Explorer anzeigen.
4. Führen Sie im Projekt-Explorer einen Rechtsklick auf den jetzt noch generischen Projektname aus und wählen Sie im Kontextmenü den Eintrag **Eigenschaften von Projekt...**.
5. Im nachfolgenden Dialog können Sie dann einen aussagekräftigen und eindeutigen Namen vergeben.

Variante in Zusammenarbeit mit einer INI-Datei

▲ Zurück zum Anfang

Wie versprochen hier noch die Variante, welche nicht mit einem Schlüssel in der Registrierdatenbank, sondern mit einer INI-Datei arbeitet.

```
Private Const Präfix = "Rechnung"
Private Const INIDatei = "c:\settings.txt"

Sub AutoNew()
  Dim oDoc As Document, oV As Template
  Set oDoc = ActiveDocument
  Set oV = oDoc.AttachedTemplate
  aSec = aSec & Left(oV.Name, InStr(oV.Name, ".") - 1)
  Num = System.PrivateProfileString(INIDatei, aSec, "Laufnummer")
  Num = Val(Num) + 1
  System.PrivateProfileString(INIDatei, aSec, "Laufnummer") = Num
  On Error Resume Next
  oDoc.CustomDocumentProperties.Add Name:="Laufnummer", _
     LinkToContent:=False, Value:=Num, Type:=msoPropertyTypeString
  On Error GoTo 0
  For Each Teil In oDoc.StoryRanges
    Teil.Fields.Update
    While Not (Teil.NextStoryRange Is Nothing)
      Set Teil = Teil.NextStoryRange
      Teil.Fields.Update
    Wend
  Next
  With Dialogs(wdDialogFileSummaryInfo)
    .Title = Präfix & Num
    .Execute
  End With
  oDoc.Windows(1).Caption = Präfix & Num
End Sub
```

(Dieser Code ist Bestandteil einer Dokumentenvorlage. Beim Generieren eines neuen Dokumentes wird eine Laufnummer vergeben. Wird das Dokument nun vom Benutzer gespeichert, wird ein Dateiname vorgeschlagen, welcher sich aus einem Präfix und einer fortlaufenden Nummer zusammen setzt. Im Gegensatz zum vorherigen Beispiel kommt hier eine INI-Datei zum Einsatz.)

Bemerkungen zum Code

- Kopieren Sie den Code im VBA-Editor in ein neues Modul der fraglichen Dokumentvorlage.
- Wenn Sie die Laufnummer im Dokument anzeigen wollen, fügen Sie der Dokumentvorlage an geeigneter Stelle eine Dokumenteigenschaft mit dem Namen **Laufnummer** hinzu, wie das Eingangs in diesem Skript beschrieben wurde.
- Sie können den **Präfix** für den Namen unter dem das Dokument gespeichert wird, jederzeit im Kopf des Codes anpassen.
- Sie können **Pfad und Name** der INI-Datei im Kopf des Codes anpassen.
- Sie können die Laufnummer auf Wunsch nachformatieren. Falls Sie eine vierstellige Zahl benötigen, welche links mit Nullen aufgefüllt ist, dann erreichen Sie dies mittels VBA wie folgt: `Num = Format(Num, "0000")`.

- Damit der Prozess bei jedem neuen Dokument aufgerufen wird, ist der Name des Prozesses, nämlich **AutoNew**, zwingend.
- Der Code ist so gestaltet, dass Sie mit eine und der selben INI-Datei beliebig viele Dokumentvorlagen abdecken können. Sie könnten die Mimik allerdings überlisten, wenn Sie Dokumentvorlagen mit dem selben Namen in verschiedenen Verzeichnissen haben und entsprechend ausrüsten. In diesem Spezialfalle sollten Sie mehrere INI-Dateien führen.

Zurücksetzen der Laufnummer

▲ Zurück zum Anfang

Sollte es einmal nötig sein, die Laufnummer (auf einen beliebigen Wert zurückzusetzen, können Sie dies mit folgendem Makro erreichen.

Die folgende Prozedur brauchen Sie nur dann, wenn Sie sich für die Variante mit der Registrierdatenbank entschieden haben. Falls Sie dagegen die Variante mit der INI-Datei gewählt haben, dann öffnen Sie die entsprechende Datei z.B. mit Notepad und setzen Sie dann den Wert hinter dem Parameter Laufnummer auf eine Zahl Ihrer Wahl.

! Kopieren Sie dieses Makro nicht irgendwo hin. **Öffnen** Sie die Dokumentvorlage, auf der die zu nummerierenden Dokumente basieren und kopieren Sie dieses Makro dort hinein. Rufen Sie es anschliessend auf. Sie werden dann nach der Startnummer gefragt.

```
Private Const Titel = "Zähler zurück setzen"
Private Const M1 = "Auf welchen Wert soll der Zähler zurück gesetzt werden?"
Private Const M2 = "Sie sollten eine Zahl eingeben."
Private Const M3 = "Der Zähler wurde zurückgesetzt."

Sub ZählerZurücksetzen()
  Dim oDoc As Document, oV As Template
  Set oDoc = ActiveDocument
  Set oV = oDoc.AttachedTemplate
  Projekt = oV.VBProject.Name
  aSec = aSec & Left(oV.Name, InStr(oV.Name, ".") - 1)
  Num = GetSetting(AppName:=Projekt, Section:=aSec, _
     Key:="Laufnummer", Default:="0")
  Flag = False
  While Flag = False
    Antwort = InputBox(M1, Titel, Num)
    If Antwort = "" Then End
    If Not IsNumeric(Antwort) Then
      MsgBox M2, vbOKOnly, Titel
    Else
      Flag = True
    End If
  Wend
  SaveSetting AppName:=Projekt, Section:=aSec, _
     Key:="Laufnummer", Setting:=Int(Antwort)
  MsgBox M3, vbOKOnly, Titel
End Sub
```

(Setzt die Laufnummer auf einen beliebigen Wert zurück. Die Routine gilt ausschliesslich für die Lösung in Zusammenarbeit mit der Registrierdatenbank.)

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Informationen für die Word-Anwendung auch über einen Windows-Neustart hinaus sichern

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Kollerat.htm

René Probst, Oktober 1999

Änderungen

- 24. November 2000 - Gestrafft wo möglich - Erweitert wo sinnvoll
- 24. November 2000 - Variante mit INI-Datei

Wie kann ich das Kreuz zum Schliessen einer UserForm in der Titelleiste unterdrücken?

◄ Zurück zum Hauptindex

Problem

> Ich habe eine kleine Anwendung geschrieben, welche auf einer *UserForm* aufbaut. Dabei ist es wichtig, dass der Benutzer diese UserForm über die Schaltfläche *Schliessen*, welche ich auf der UserForm vorsehe, beendet. Schliesst der Benutzer nun die UserForm aber trotzdem über das Kreuz in der Titelleiste, kann ich die Daten auf der *UserForm* nicht zurückschreiben. Sie gehen damit verloren. Ich möchte dieses Kreuz also zum verschwinden bringen, bzw. deaktivieren.

Antwort

Sie können die Titelleiste oder dieses Kreuz nicht mit einfachen Mitteln zum Verschwinden bringen. Dazu müsste ein Windows-API herhalten. Zudem wäre die Sache auch dann nicht wasserdicht, kann der Benutzer die UserForm doch immer noch über die Tastenkombination **[Alt+F4]** schliessen.

Eine weitere Möglichkeit eine UserForm zu schliessen bietet übrigens auch die **[Esc]**-Taste, wenn Sie dies in der UserForm vorsehen und bei einer der Befehlsschaltflächen die Eigenschaft `Cancel = True` setzen.

Gefragt ist also ein besserer Mechanismus. Diese Mimik ist letztlich auch an Bord und nennte sich **QueryClose**-Ereignis.

Das *QueryClose*-Ereignis liefert ein Flag zurück, welches angibt, ob die UserForm auf Grund eines expliziten *Unload*-Befehles oder durch eine Intervention des Benutzers geschlossen wird.

Dank dieser Mimik, müssen wir eigentlich nicht einmal mehr genau wissen, welche Varianten es gibt, um eine UserForm zu schliessen. Trotzdem sollen diese Varianten hier nochmals aufgeführt werden.

- Explizit mittels *Unload*-Anweisung im Code. Dabei wird das *CloseMode-Flag* **eingeschaltet**.
- Implizit durch den Benutzer, wenn er das Schliessen-Kreuz anklickt. Dabei bleibt das *CloseMode-Flag* **ausgeschaltet**.
- Implizit durch den Benutzer, wenn er die Tastenkombination **[Alt+F4]** betätigt. Dabei bleibt das *CloseMode-Flag* **ausgeschaltet**.
- Implizit durch den Benutzer, wenn er die **[Esc]** Taste drückt und in unserem Code eine Befehlsschaltfläche vorgesehen ist, welche die Eigenschaft **Cancel = True** gesetzt hat.

Lösungen

Drei Szenarien

Im nachfolgenden werden drei in der Praxis typische Szenarien durchgespielt. Im ersten Beispiel wird das Schliessen über das Kreuz (und ähnliche Interventionen) schlicht unterbunden. Im zweiten Beispiel geht eine Warnung an den Benutzer und im letzten Beispiel wird diese Intervention des Benutzers genau gleich behandelt, als hätte er die vom Programmierer zur Verfügung gestellte Schaltfläche *Speichern/Schliessen* betätigt.

Vorbereitungen

Um diese Szenarien durchzuspielen richten wir uns eine einfache UserForm ein, welche folgende Elemente beherbergt:

- Ein Textfeld mit den Namen **Textbox1**.
- Eine Befehlsschaltfläche mit den Namen **CommandButton1** und der Bezeichnung *Speichern/Schliessen*.

VBA-Code, welches das Schliessen über das Kreuz verwirft

Im folgenden Beispiel, geschieht faktisch nichts, wenn der Benutzer das Schliessen-Kreuz betätigt. Die UserForm bleibt also weiterhin aktiv.

Wenn Sie diese Methode anwenden, ist es natürlich vital, dass Sie dem Benutzer eine faire Alternative zum Schliessen der UserForm anbieten. Im Beispiel wird dies mit dem Knopf *CommandButton1* sichergestellt.

Sie können den Code in Schritten abbauen und werden dann feststellen, dass das *QueryClose-Ereignis* aufgerufen wird, wenn der Benutzer das Schliessen-Kreuz anklickt und wenn der Benutzer die Schaltfläche betätigt. Im letzteren Fall erfolgt dies als Folge der *Unload*-Anweisung. Der wichtige Unterschied liegt darin, dass im ersten Fall das *CloseMode-Flag* nicht gesetzt ist im zweiten Fall aber schon.

Mit dem *Cancel-Flag*, welches im Ereignis gesetzt wird, wird dann die Aktion des Benutzers verworfen.

Kopieren Sie den folgenden Code in die vorbereitete UserForm und testen Sie dann das Verhalten des Formulars.

```
Private Const Anw = "MeineAnwendung"
Private Const Abschnitt = "MeinAbschnitt"

Private Sub UserForm_Initialize()
  'Inhalt des Textfeldes zurückholen
  TextBox1.Text = GetSetting(AppName:=Anw, Section:=Abschnitt, _
     Key:="TextBox1", Default:="Name des Fluggastes")
End Sub

Private Sub CommandButton1_Click()
  'Inhalt des Textfeldes zurückschreiben
  SaveSetting AppName:=Anw, Section:=Abschnitt, _
     Key:="TextBox1", Setting:=TextBox1.Text
  Unload Me ' Userform explizit beenden
End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If CloseMode <> 1 Then Cancel = True
End Sub
```

(Mit diesem Code wird das Schliessen-Kreuz faktisch deaktiviert.)

VBA-Code, mit Sicherheitsabfrage beim Schliessen der UserForm

Im nächsten Beispiel, kann der Benutzer das Formular über das Kreuz schliessen. Allerdings erhält er dann zuerst eine Sicherheitsabfrage und der Benutzer kann dann bestimmen, was nun geschehen oder eben nicht geschehen soll.

Kopieren Sie den folgenden Code in die vorbereitete UserForm und testen Sie dann das Verhalten des Formulars.

```
Private Const Anw = "MeineAnwendung"
Private Const Abschnitt = "MeinAbschnitt"

Private Sub UserForm_Initialize()
  'Inhalt des Textfeldes zurückholen
  TextBox1.Text = GetSetting(AppName:=Anw, Section:=Abschnitt, _
     Key:="TextBox1", Default:="Name des Fluggastes")
End Sub

Private Sub CommandButton1_Click()
  'Inhalt des Textfeldes zurückschreiben
  SaveSetting AppName:=Anw, Section:=Abschnitt, _
     Key:="TextBox1", Setting:=TextBox1.Text
  Unload Me 'Userform explizit beenden
End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If CloseMode = 0 Then
    Select Case MsgBox("Wollen Sie die Daten verwerfen?", vbQuestion + vbYesNoCancel)
      Case vbYes 'Ja
        Unload Me
      Case vbNo 'Nein
        CommandButton1_Click
      Case vbCancel 'Abbrechen
        Cancel = True
    End Select
  End If
End Sub
```

(Bevor die UserForm schliesst, erfolgt eine Sicherheitsabfrage beim Benutzer.)

VBA-Code, welcher Speichern/Schliessen emuliert

Schliesslich noch der Code, welcher die Daten unbedingt zurückschreibt, wenn der Benutzer das Schliessen-Kreuz anklickt.

Kopieren Sie den folgenden Code in die vorbereitete UserForm und testen Sie dann das Verhalten des Formulars.

```
Private Const Anw = "MeineAnwendung"
Private Const Abschnitt = "MeinAbschnitt"

Private Sub UserForm_Initialize()
  'Inhalt des Textfeldes zurückholen
  TextBox1.Text = GetSetting(AppName:=Anw, Section:=Abschnitt, _
     Key:="TextBox1", Default:="Name des Fluggastes")
End Sub

Private Sub CommandButton1_Click()
  'Inhalt des Textfeldes zurückschreiben
  SaveSetting AppName:=Anw, Section:=Abschnitt, _
     Key:="TextBox1", Setting:=TextBox1.Text
  Unload Me 'Userform explizit beenden
End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If CloseMode = 0 Then CommandButton1_Click
End Sub
```

(Auch wenn der Benutzer das Schliessen-Kreuz anklickt, werden die Daten forciert zurück geschrieben, bevor das Formular dann schliesst.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/QueryCls.htm

René Probst, November 2000

Änderungen

- --- nil ---

Benutzerspezifische Standardwerte für konventionelle Formularfelder, Textmarkentext und UserForm-Controls

◄ Zurück zum Hauptindex

Diese Seite

- ▼ Aufgabenstellungen
- ▼ Lösungsvarianten
- ▼ Konventionelle Textformularfelder mit benutzerspezifischen Standardwerten belegen
- ▼ Dropdown-Formularfeld aus einer INI-Datei laden
- ▼ Textmarkentext mit benutzerspezifischen Standardwerten belegen
- ▼ Arten von Textmarken und ihr Verhalten
- ▼ Text-,und Optionsfelder sowie Kontrollkästchen auf einer UserForm mit benutzerspezifischen Standardwerten belegen
- ▼ Kombinationsfeld (ComboBox) aus einer INI-Datei laden
- ▼ Aus einer Excel-Tabelle eine INI-Datei generieren
- ▼ Links zu diesem Thema

Aufgabenstellungen

▲ Zurück zum Anfang

> 1. Ich habe eine Dokumentenvorlage mit konventionellen Formularfeldern in Word erstellt. Einige dieser Formulartextfelder möchte ich beim generieren eines Formulars automatisch ausfüllen, wobei die Werte vom Benutzer, welcher das Formular erstellt, abhängig sein sollten.
>
> 2. Ich habe eine Dokumentenvorlage mit Textmarken in Word erstellt. Einige dieser Textmarken möchte ich beim generieren eines neuen Dokuments automatisch ausfüllen, wobei die Werte vom Benutzer, welcher das Dokument erstellt, abhängig sein sollten.
>
> 3. Ich habe eine UserForm mit einer stattlichen Anzahl von Schaltflächen eingerichtet. Einige Textfelder, Kontrollkästchen und Optionsfelder auf dieser UserForm möchte ich beim Anzeigen der UserForm automatisch ausfüllen, wobei die Werte vom Benutzer, welcher am PC arbeitet, abhängig sein sollten.

Lösungsvarianten

▲ Zurück zum Anfang

MS Word bietet dazu mindestens zwei flexible Handhaben und zwar im einzelnen:

- Unter Verwendung einer sogenannten **INI-Datei**. Diese Methode wird im anschliessenden Text ausführlich beschrieben. Die Methode ist einfach zu handhaben und eigentlich sehr transparent.
- Unter Verwendung von Einträgen in der **Registrierdatenbank**. Diese Methode wird hier nicht beschrieben, weil sie der ersten Methode doch sehr ähnelt. Bei Bedarf erhalten Sie dazu aber Informationen in den beiden Links, welche am Ende dieses Skriptes eingeblendet sind.

Konventionelle Textformularfelder mit benutzerspezifischen Standardwerten belegen

▲ Zurück zum Anfang

Testumgebung

Um das System zu verstehen machen Sie bitte folgendes Experiment:

1. Erstellen Sie eine neue Dokumentenvorlage.
2. Lassen Sie sich über den Word-Dialog ***Extras - Anpassen*** die Symbolleiste mit dem Namen **Formular** anzeigen.
3. Plazieren Sie **drei Textformularfelder** im Dokument.
4. Führen Sie einen Doppelklick auf das erste Formulartextfeld aus.
5. Tragen Sie im Eigenschaftsdialog des Textfeldes unter *Textmarke* den Namen **Vorname** ein.
6. Führen Sie einen Doppelklick auf das zweite Formulartextfeld aus.
7. Tragen Sie im Eigenschaftsdialog des Textfeldes unter *Textmarke* den Namen **Nachname** ein.
8. Führen Sie einen Doppelklick auf das dritte Formulartextfeld aus.
9. Trage im Eigenschaftsdialog des Textfeldes unter Textmarke den Namen **PIN** ein.
10. Wechseln sie nun mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
11. Im VBA-Editor wählen Sie nun vom Hauptmenü den Befehl ***Einfügen - Modul.***
12. Kopieren Sie nun den unten abgedruckten Code in das leere Fenster. Der Name **AutoNew** des Prozesses ist hier zwingend, damit der Code beim Erstellen eines neuen Formulars automatisch anläuft.
13. Kehren Sie nun zum Dokument-Fenster zurück, **schützen** und speichern Sie die Vorlage.
14. Rufen Sie nun das kleine Programm **Notepad.exe** (Start - Programme - Zubehör) auf und erstellen Sie damit eine Datei mit den paar wenigen Zeilen, welche unten abgedruckt sind.
15. Speichern Sie diese Datei unter dem Pfad/Namen: **C:\Eigene Dateien\Benutzer.ini**. Sie dürfen der Datei auch einen anderen Namen vergeben oder die Datei in einem anderen Pfad abspeichern, müssten aber in diesem Falle den Wert für die Konstante **Pfad** im Kopf des VBA-Codes anpassen.

Die INI-Datei

Die INI-Datei ist sehr einfach aufgebaut und sieht wie folgt aus:

```
[Formular1]
Vorname=James
Nachname=Bond
PIN=007
```

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\Benutzer.ini" 'Pfad und Name der INI-Datei
Private Const Abschnitt = "Formular1" 'Name des Abs. in der INI-Datei oder *Vorlage

Sub AutoNew()
  If Abschnitt = "*Vorlage" Then
    Sektion = ActiveDocument.AttachedTemplate
  Else
    Sektion = Abschnitt
  End If
  chk = Dir(Pfad)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & Pfad & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  Dim oDoc As Document
  Dim FF As FormField
  Set oDoc = ActiveDocument
  For Each FF In oDoc.FormFields
    x = System.PrivateProfileString(FileName:=Pfad, _
       Section:=Sektion, Key:=FF.Name)
    If x <> "" Then FF.Result = x
  Next
End Sub
```

(Die Werte, welche in der INI-Datei gespeichert sind, werden beim Erstellen eines neuen Formulars automatisch in die Formularfelder eingetragen.)

Anmerkungen zum Verfahren

- Es gibt auch eine Word-integrierte Möglichkeit, einen Standardwert für ein Textformularfeld vorzugeben. Diese Funktion finden Sie im Eigenschaftsdialog (Optionen) jedes Textformularfeldes. Sie können damit aber keine benutzerspezifische Werte einstellen und das Administrieren der Werte erweist sich damit doch etwas umständlich.
- Sie können eine (fast) beliebige Anzahl von Felder mit (fast) beliebigen Feldnamen in der oben gezeigten Art und Weise abhandeln und zwar ohne, dass der Code angepasst werden muss. Vital ist nur, dass die Feldnamen mit den Parametern in der INI-Datei übereinstimmen und zwar buchstäblich und exakt.
- Sie können problemlos in der selben INI-Datei Standardwerte für verschiedene Formulare aufnehmen, dabei dürfen sich die Feldnamen in einem anderen Formular durchaus wiederholen. Wählen Sie als Abschnittsname für ein weiteres Formular einfach z.B. **[Formular2}**, kopieren Sie den oben abgedruckten Code auch in diese Formularvorlage und vergessen Sie nicht im Kopf des Codes die Konstante **Abschnitt** entsprechend anzupassen. Dies sind dann schon alle nötigen Vorkehrungen, welche Sie in diesem Falle treffen müssen.
- In manchen Fällen, ist es auch eine gute Idee, diesen Abschnitt nach dem Namen der Formularvorlage zu benennen. Wenn Sie dies tun möchten, dann setzen Sie im Kopf des Codes die Konstante mit dem Namen Abschnitt auf den symbolischen Wert ***Vorlage**. Damit wird der Code noch universeller. Angenommen Ihre Formularvorlage heisst *Rechnung.dot*, dann tragen sie in der INI-Datei die Standardwerte im Abschnitt **[Rechnung.dot]** ein.

- Die oben präsentierte Lösung geht davon aus, dass Sie für jeden Benutzer in Ihrer Organisation eine eigene INI-Datei einrichten, einrichten lassen oder vertreiben. Es mag für Sie mehr Sinn machen, die Standardwerte für alle Benutzer in einer einzigen INI-Datei zu administrieren. Diese Variante wird gleich anschliessend besprochen.

- Ihnen graut davor, für vielleicht 400 Benutzer und Benutzerinnen eine INI-Datei, welche doch ein etwas unpraktisches Format aufweist, zu führen? Kein Problem, später in diesem Skript wird besprochen, wie Sie die Standardwerte für Ihre Benutzer in Excel administrieren können. Auch eine Excel-Routine, welche es erlaubt auf Knopfdruck aus einer Excel-Tabelle eine INI-Datei zu generieren finden Sie weiter unten, zusammenfassend für alle Methoden, welche in diesem Skript besprochen werden.

Dropdown-Formularfeld aus einer INI-Datei laden

▲ Zurück zum Anfang

Im folgenden erfahren Sie, wie eine **Dropdown-Formularfeld**, welches bis zu 25 Einträge dem Benutzer zur Disposition stellt, ex INI-Datei geladen werden kann.

Die INI-Datei

Um die Sache zu testen erstellen Sie bitte eine Datei, welchen folgenden Inhalt aufweist:

```
[Elemente]
L1=Wasserstoff
L2=Helium
L3=Lithium
L4=Beryllium
L5=Bor
L6=Kohlenstoff
L7=Stickstoff
L8=Sauerstoff
L9=Flor
L10=Neon
L11=Natrium
L12=Magnesium
```

Der Code um das Dropdown-Formularfeld mit dem Namen **Dropdown1** zu laden lautet nun wie folgt:

VBA-Code

```
Private Const INIDatei = "C:\Elemente.ini"

Sub DropdownFeldLaden()
  chk = Dir(INIDatei)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & INIDatei & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  i = 1
  x = DropdownFeldLadenEintrag("L" & i)
  While Not x = ""
    ActiveDocument.FormFields("Dropdown1").DropDown.ListEntries.Add x
    i = i + 1
    x = DropdownFeldLadenEintrag("L" & i)
  Wend
End Sub

Private Function DropdownFeldLadenEintrag(Key As String) As String
  DropdownFeldLadenEintrag = _
     System.PrivateProfileString(FileName:=INIDatei, _
        Section:="Elemente", Key:=Key)
End Function
```

(Das Dropdown-Formularfeld mit dem Namen *Dropdown1* wird ex INI-Datei geladen.)

Anmerkung

! Wie schon erwähnt, ist ein Dropdown-Formularfeld auf 25 Listeneinträge limitiert. Diese Beschränkung hat mit dem Aussehen der Schaltfläche zu tun, erlaubt diese doch keine *Scrolling*. Somit ist der Bildschirm irgendwann voll, je nach Auflösung sogar ziemlich rasch. Auf der Datenbank von Microsoft finden Sie eine Umgehungsmöglichkeit, welche in Verbindung mit einer *UserForm* arbeitet. Den Link dazu finden Sie am Ende dieses Skriptes.

Eine INI-Datei für alle Benutzer und Benutzerinnen in Ihrer Organisation

Sollten Sie es vorziehen, die Standardwerte für ein bestimmtes Formular für alle Benutzer und Benutzerinnen in einer INI-Datei zu führen, dann muss das Programm als erstes erkennen, wer nun am Rechner sitzt. Als Schlüssel dazu eignet sich insbesondere die **Windows NT-Kennung** unter der sich die Person bei Windows anmeldet. Alternativ könnte man auch den Wert heranziehen, welche man in Word im Dialog ***Extras - Optionen - Benutzerinformationen - Initialen:*** oder ***Name:*** vorfindet. Die nachfolgende Routine deckt jedenfalls diese drei Möglichkeiten ab. Die Methode mit der Windows-Kennung sollte auch unter **Windows 98** funktionieren, falls Sie den Rechner als Mehrplatzrechner einsetzen.

```
'*** Beginn des anpassbaren Teiles
Private Const Pfad = "C:\Eigene Dateien\Benutzer.ini" 'Pfad und Name der INI-Datei
Private Const Kennung = "NT-Kennung" 'NT-Kennung; Word-Initialen; Word-Name
'*** Ende des anpassbaren Teiles

Declare Function WNetGetUser Lib "mpr.dll" _
  Alias "WNetGetUserA" (ByVal lpName As String, _
     ByVal lpUserName As String, lpnLength As Long) As Long
Private Const NoError = 0

Sub AutoNew()
  chk = Dir(Pfad)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & Pfad & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  Select Case Kennung
    Case "Word-Initialen"
      Sektion = Application.UserInitials
    Case "Word-Name"
      Sektion = Application.UserName
    Case Else
      Sektion = NTKennung
  End Select
  Dim oDoc As Document
  Dim FF As FormField
  Set oDoc = ActiveDocument
  For Each FF In oDoc.FormFields
    x = System.PrivateProfileString(FileName:=Pfad, _
       Section:=Sektion, Key:=FF.Name)
    If x <> "" Then FF.Result = x
  Next
End Sub

Private Function NTKennung()
  Const lpnLength As Integer = 255
  Dim status As Integer
  Dim lpName, lpUserName As String
  lpUserName = Space$(lpnLength + 1)
  status = WNetGetUser(lpName, lpUserName, lpnLength)
  If status = NoError Then
    lpUserName = Left$(lpUserName, InStr(lpUserName, Chr(0)) - 1)
  End If
  NTKennung = lpUserName
End Function
```

(Mit einem Windows-API wird die NT-Kennung des Benutzers in Erfahrung gebracht. Anschliessend werden die Standardwerte dieses spezifischen Benutzers geladen und in die Formularfelder eingetragen. Diese Routine ermöglicht es, alle Benutzer in Ihrer Organisation in einer einzigen INI-Datei zu führen.)

Anmerkungen

Hier nochmals in Kürze, was die Mimik bezwecken will und was von Ihnen erwartet wird:

1. Setzen Sie den Wert der Konstante **Pfad** auf den tatsächlichen Namen Ihrer INI-Datei.
2. Entscheiden Sie sich, ob der Benutzer auf Grund seiner NT-Kennung, seinen Benutzerinitialen in der Word-Anwendung oder seines Benutzername in der Word-Anwendung erkannt werden soll. Entsprechend Ihrer Wahl setzen Sie den Wert der Konstante **Kennung** auf einen der folgenden möglichen Werte:
   - NT-Kennung
   - Word-Initialen
   - Word-Name
3. Die Abschnitte in der INI-Datei, müssen nun gleich lauten wie die NT-Kennung, die Word-Benutzerinitialen oder der Word-Bemutzername. Für eine volle Abdeckung sollte es also für jeden Benutzer in der Organisation einen Abschnitt im folgenden Sinne geben: **[HMeier]**.
4. Kopieren Sie dann den oben abgedruckten Code in die Formularvorlage.

Ihnen graut davor, für vielleicht 400 Benutzer und Benutzerinnen eine INI-Datei, welche doch ein etwas unpraktisches Format aufweist, zu führen? Kein Problem, später in diesem Skript wird besprochen, wie Sie Die Standardwerte für Ihre Benutzer in Excel administrieren können. Auch eine Excel-Routine, welche es erlaubt auf Knopfdruck aus einer Excel-Tabelle eine INI-Datei zu generieren finden Sie weiter unten, zusammenfassend für alle Methoden, welche in diesem Skript besprochen werden.

Textmarkentext mit benutzerspezifischen Standardwerten belegen

▲ Zurück zum Anfang

Testumgebung

Um das System zu verstehen machen Sie bitte folgendes Experiment:

1. Erstellen Sie eine neue Dokumentenvorlage.
2. Plazieren Sie **drei Textmarken** im Dokument. Diese Textmarken sollen die Namen ***Vorname***, ***Nachname*** und ***PIN*** erhalten.
3. Wechseln sie nun mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
4. Im VBA-Editor wählen Sie nun vom Hauptmenü den Befehl ***Einfügen - Modul.***
5. Kopieren Sie nun den unten abgedruckten Code in das leere Fenster. Der Name **AutoNew** des Prozesses ist hier zwingend, damit der Code beim Erstellen eines neuen Dokumentes automatisch anläuft.
6. Speichern Sie die Vorlage nun ab.
7. Rufen Sie nun das kleine Programm **Notepad.exe** (Start - Programme - Zubehör) auf und erstellen Sie damit eine Datei mit den paar wenigen Zeilen, welche unten abgedruckt sind.
8. Speichern Sie diese Datei unter dem Pfad/Namen: **C:\Eigene Dateien\Benutzer.ini**. Sie dürfen der Datei auch einen anderen Namen vergeben oder die Datei in einem anderen Pfad abspeichern, müssten aber in diesem Falle den Wert für die Konstante **Pfad** im Kopf des VBA-Codes anpassen.

Die INI-Datei

Die INI-Datei ist sehr einfach aufgebaut und sieht wie folgt aus:

```
[Geschäftsbrief.dot]
Vorname=James
Nachname=Bond
PIN=007
```

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\Benutzer.ini"
Private Const Abschnitt = "Geschäftsbrief.dot"

Sub AutoNew()
  chk = Dir(Pfad)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & Pfad & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  Dim oDoc As Document
  Dim TM As Bookmark
  Set oDoc = ActiveDocument
  For Each TM In oDoc.Bookmarks
    x = System.PrivateProfileString(FileName:=Pfad, _
       Section:=Abschnitt, Key:=TM.Name)
    If x <> "" Then TM.Range.Text = x
  Next
End Sub
```

(Die Werte, welche in der INI-Datei gespeichert sind, werden beim Erstellen eines neuen Dokumentes automatisch bei den Textmarken eingetragen.)

Anmerkungen zum Verfahren

- Eigentlich kennt Word zwei verschiedene Arten von Textmarken. Dieser Unterschied wird anschliessend noch ausführlich erklärt.
- Sie können eine (fast) beliebige Anzahl von Textmarken mit (fast) beliebigen Textmarkennamen in dieser Art und Weise abhandeln und zwar ohne, dass der Code angepasst werden muss. Vital ist nur, dass die Feldnamen mit den Parametern in der INI-Datei übereinstimmen und zwar buchstäblich und exakt.
- Sie können problemlos in der selben INI-Datei Standardwerte für Textmarken in verschiedenen Dokumentenvorlagen aufnehmen. Dabei dürfen sich die Textmarkennamen in einer anderen Vorlagen durchaus wiederholen. Wählen Sie als Abschnittsname für eine weitere Dokumentenvorlage einfach z.B. **[Rechnung.dot}** (an Stelle von [Geschäftsbrief.dot]). Kopieren Sie den oben abgedruckten Code auch in diese Dokumentenvorlage und vergessen Sie nicht im Kopf des Codes die Konstante **Abschnitt** entsprechend anzupassen. Dies sind dann schon alle nötigen Vorkehrungen, welche Sie in diesem Falle treffen müssen.
- In manchen Fällen, ist es eine gute Idee, diesen Abschnitt nach dem Namen der Dokumentenvorlage zu benennen. Wenn Sie dies tun möchten, dann setzen Sie im Kopf des Codes die Konstante mit dem Namen Abschnitt auf den symbolischen Wert ***Vorlage**. Damit wird der Code noch universeller. Angenommen Ihre Dokumentenvorlage heisst *Rechnung.dot*, dann tragen sie in der INI-Datei die Standardwerte im Abschnitt **[Rechnung.dot]** ein.
- Die oben präsentierte Lösung geht davon aus, dass Sie für jeden Benutzer in Ihrer Organisation eine eigene INI-Datei einrichten. einrichten lassen, bzw. vertreiben. Es mag für Sie mehr Sinn machen, die Standardwerte für alle Benutzer in einer einzigen INI-Datei zu administrieren. Diese Variante wird etwas später in diesem Dokument besprochen.

☛ Ihnen graut davor, für vielleicht 400 Benutzer und Benutzerinnen eine INI-Datei, welche doch ein etwas unpraktisches Format aufweist, zu führen? Kein Problem, später in diesem Skript wird besprochen, wie Sie die Standardwerte für Ihre Benutzer in Excel administrieren können. Auch eine Excel-Routine, welche es erlaubt auf Knopfdruck aus einer Excel-Tabelle eine INI-Datei zu generieren finden Sie weiter unten.

Arten von Textmarken und ihr Verhalten

▲ Zurück zum Anfang

Was viele zu wenig beachten, Word kennt eigentlich zwei Arten von Textmarken, nämlich offene und geschlossenen. Um dies zu erkennen, machen wir folgenden Test:

Offene Textmarken

Damit Textmarken überhaupt am Bildschirm angezeigt werden, sollten Sie zuerst die Einstellung ***Extras - Optionen - Ansicht - Anzeigen - Textmarken*** überprüfen.

Setzen sie die Einfügemarke im Dokument an die gewünschte Stelle, und wählen Sie **ohne dass etwas im Dokument markiert wäre**, den Dialog ***Einfügen - Textmarke...*** und vergeben Sie der Marke einen Namen. Das Resultat ist eine **offene Textmarke** und wird wie folgt angezeigt: `[`. Mit diesem Vorgehen, haben Sie letztlich eine **Position** im Dokument definiert.

Weisen Sie mittels VBA bei dieser Art von Textmarke einen Text zu, also z.B. *Hallo Welt* und Sie lassen diesen Code zehn mal ablaufen, dann steht dort 10x Hallo Welt. Der Text nach der Textmarke wird dabei immer nach recht geschoben. Die Textmarke selbst ist dabei nicht betroffen, sie bleibt erhalten an der selben Stelle. Die Textmarke repräsentiert ja eine Position und eine Position kann man physisch nicht löschen.

Geschlossene Textmarken

Als nächstes markieren Sie nun einen kurzen Text z.B. ein Wort im Dokument. und wählen dann wieder den Dialog ***Einfügen - Textmarke...*** dann vergeben Sie der Marke einen Namen. Das Resultat ist eine **geschlossene Textmarke** und wird wie folgt angezeigt: `[ ]`. Sie haben letztlich einen **Bereich** definiert. Offiziell wird diese Art von Textmarken übrigens **Textmarkenfeld** genannt.

Weisen Sie mittels VBA bei dieser Art der Textmarke einen Text zu, also z.B. Hallo Welt und Sie lassen diesen Code zum zweiten Mal ablaufen, dann wird es nicht mehr klappen, viel mehr werden Sie eine Fehlermeldung erhalten. Der Bereich wurde nämlich beim ersten Durchlauf überschrieben und zwar inklusive Textmarke.

Dieses Verhalten ist oft praxisgerecht, ja sogar erwünscht; allerdings nicht immer. Das nächste Kapitel zeigt ein dynamisches System auf, also eine Textmarke, deren Textmarkentext ich beliebig oft manipulieren kann. Dabei wird der bestehende Text mit dem neuen Text überschrieben, die Textmarke bleibt dabei aber faktisch erhalten. Eigentlich wird mit einem Trick gearbeitet: die Textmarke wird einfach am Schluss wieder bei der ursprünglichen Stelle gesetzt. Deshalb habe ich diese Mimik ***Recycling Textmarke*** getauft.

Recycling Textmarke ex VBA

Damit das folgende Experiment gelingt muss eine **geschlossene Textmarke** mit dem Namen **MeineMarke** definiert werden. Was innerhalb der geschlossenen Textmarke steht, ist völlig irrelevant. Wichtig ist der Name der Marke, welcher mit der Syntax im Code korrespondieren muss.

```
Sub RecyclingTextmarke()
  TMName = "MeineMarke"
  If ActiveDocument.Bookmarks.Exists(TMName) Then
    Dim TMRange As Range
    Set TMRange = ActiveDocument.Bookmarks(TMName).Range
    TMRange.Text = Time
    ActiveDocument.Bookmarks.Add Name:=TMName, Range:=TMRange
  End If
End Sub
```

(Das *Recycling* einer geschlossenen Textmarke.)

Hier also nochmals, die selbe Routine, wie die obige. Diesmal kommen aber statt offene Textmarken, *Recycling-Textmarken* zum Einsatz:

```
Private Const Pfad = "C:\Eigene Dateien\Benutzer.ini"
Private Const Abschnitt = "Geschäftsbrief.dot"

Sub AutoNew()
  chk = Dir(Pfad)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & Pfad & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  Dim oDoc As Document
  Dim TM As Bookmark
  Dim TMRange As Range
  Set oDoc = ActiveDocument
  For i = 1 To oDoc.Bookmarks.Count
    Set TM = oDoc.Bookmarks(i)
    x = System.PrivateProfileString(FileName:=Pfad, _
       Section:=Abschnitt, Key:=TM.Name)
    If x <> "" Then
      Set TMRange = TM.Range
      TMName = TM.Name
      TMRange.Text = x
      oDoc.Bookmarks.Add Name:=TMName, Range:=TMRange
    End If
  Next i
End Sub
```

(Die Werte, welche in der INI-Datei gespeichert sind, werden beim Erstellen eines neuen Dokumentes automatisch bei den *Recycling-Textmarken* eingetragen.)

Eine INI-Datei für alle Benutzer und Benutzerinnen in Ihrer Organisation

Sollten Sie es vorziehen, die Standardwerte der Textmarken für eine bestimmte Dokumentenvorlage für alle Benutzer und Benutzerinnen in einer einzigen INI-Datei zu führen, dann muss das Programm als erstes erkennen, wer nun am Rechner sitzt. Als Schlüssel dazu eignet sich insbesondere die **Win**dows NT-**Kennung** unter der sich die Person bei Windows anmeldet. Alternativ könnte man auch den Wert heranziehen, welche man in Word im Dialog ***Extras - Optionen - Benutzerinformationen - Initialen:*** oder ***Name:*** vorfindet. Die nachfolgenden Routinen decken jedenfalls diese drei Möglichkeiten ab. Die Methode mit der Windows-Kennung sollte auch unter **Windows 98** funktionieren, falls Sie den Rechner als Mehrplatzrechner einsetzen.

Sie finden nun zuerst ein Beispiel, wie die INI-Datei dazu aussehen sollte. Daran anschliessend finden Sie den VBA-Code für die Variante mit offenen Textmarken und nochmals anschliessend die Lösung für Recycling-Textmarken.

Die INI-Datei

Angenommen, Ihre Organisation umfasst drei Mitarbeiter, welche die NT-Kennung *jwg*, *hwm* und *hjs* besitzen und die Textmarkennamen im Dokument lauten *Nachname*, *Vorname* und *PIN*, dann sieht die INI-Datei wie folgt aus:

```
[jwg]
Vorname=Wolfgang
Nachname=Goethe
PIN=0001

[hwm]
Vorname=Hans
Nachname=Meier
PIN=0002

[fjs]
Vorname=Josef
Nachname=Strauss
PIN=0003
```

Code für offene Textmarken

```
'*** Beginn des anpassbaren Teiles
Private Const Pfad = "C:\Eigene Dateien\Benutzer.ini"
Private Const Kennung = "NT-Kennung" 'NT-Kennung; Word-Initialen; Word-Name
'*** Ende des anpassbaren Teiles

Declare Function WNetGetUser Lib "mpr.dll" _
  Alias "WNetGetUserA" (ByVal lpName As String, _
     ByVal lpUserName As String, lpnLength As Long) As Long
Private Const NoError = 0

Sub AutoNew()
  chk = Dir(Pfad)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & Pfad & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  Select Case Kennung
    Case "Word-Initialen"
      Sektion = Application.UserInitials
    Case "Word-Name"
      Sektion = Application.UserName
    Case Else
      Sektion = NTKennung
  End Select
  Dim oDoc As Document
  Dim TM As Bookmark
  Set oDoc = ActiveDocument
  For Each TM In oDoc.Bookmarks
    x = System.PrivateProfileString(FileName:=Pfad, _
       Section:=Sektion, Key:=TM.Name)
    If x <> "" Then TM.Range.Text = x
  Next
End Sub

Private Function NTKennung()
  Const lpnLength As Integer = 255
  Dim status As Integer
  Dim lpName, lpUserName As String
  lpUserName = Space$(lpnLength + 1)
  status = WNetGetUser(lpName, lpUserName, lpnLength)
  If status = NoError Then
    lpUserName = Left$(lpUserName, InStr(lpUserName, Chr(0)) - 1)
  End If
  NTKennung = lpUserName
End Function
```

(Sollten Sie es vorziehen, eine einzige INI-Datei für alle Benutzer in Ihrer Organisation zu führen, kann die oben abgedruckte Routine zur Anwendung kommen. Diese Routine arbeitet mit offenen Textmarken.)

Code für *Recycling-Textmarken*

```
'*** Beginn des anpassbaren Teiles
Private Const Pfad = "C:\Eigene Dateien\Benutzer.ini"
Private Const Kennung = "NT-Kennung" 'NT-Kennung; Word-Initialen; Word-Name
'*** Ende des anpassbaren Teiles

Declare Function WNetGetUser Lib "mpr.dll" _
  Alias "WNetGetUserA" (ByVal lpName As String, _
     ByVal lpUserName As String, lpnLength As Long) As Long
Private Const NoError = 0

Sub AutoNew()
  chk = Dir(Pfad)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & Pfad & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  Select Case Kennung
    Case "Word-Initialen"
      Sektion = Application.UserInitials
    Case "Word-Name"
      Sektion = Application.UserName
    Case Else
      Sektion = NTKennung
  End Select
  Dim oDoc As Document
  Dim TM As Bookmark
  Dim TMRange As Range
  Set oDoc = ActiveDocument
  For i = 1 To oDoc.Bookmarks.Count
    Set TM = oDoc.Bookmarks(i)
    x = System.PrivateProfileString(FileName:=Pfad, _
       Section:=Sektion, Key:=TM.Name)
    If x <> "" Then
      Set TMRange = TM.Range
      TMName = TM.Name
      TMRange.Text = x
      oDoc.Bookmarks.Add Name:=TMName, Range:=TMRange
    End If
  Next i
End Sub

Private Function NTKennung()
  Const lpnLength As Integer = 255
  Dim status As Integer
  Dim lpName, lpUserName As String
  lpUserName = Space$(lpnLength + 1)
  status = WNetGetUser(lpName, lpUserName, lpnLength)
  If status = NoError Then
    lpUserName = Left$(lpUserName, InStr(lpUserName, Chr(0)) - 1)
  End If
  NTKennung = lpUserName
End Function
```

(Sollten Sie es vorziehen, eine einzige INI-Datei für alle Benutzer in Ihrer Organisation zu führen, kann die oben abgedruckte Routine zur Anwendung kommen. Diese Routine arbeitet mit *Recycling-Textmarken* .)

Anmerkungen

Hier nochmals in Kürze, was die Mimik bezwecken will und was von Ihnen erwartet wird:

1. Setzen Sie den Wert der Konstante **Pfad** auf den tatsächlichen Namen Ihrer INI-Datei.
2. Entscheiden Sie, ob der Benutzer auf Grund seiner NT-Kennung, seinen Benutzerinitialen in der Word-Anwendung oder seines Benutzername in der Word-Anwendung erkannt werden soll. Entsprechend Ihrer Wahl setzen Sie den Wert der Konstante **Kennung** auf einen der folgenden möglichen Werte:
   - NT-Kennung
   - Word-Initialen
   - Word-Name
3. Die Abschnitte in der INI-Datei, müssen nun gleich lauten wie die NT-Kennung, die Word-Benutzerinitialen oder der Word-Benutzername. Für eine volle Abdeckung sollte es also für jeden Benutzer in der Organisation einen Abschnitt im folgenden Sinne geben: **[HMeier]**.
4. Kopieren Sie dann den oben abgedruckten Code in die Dokumentvorlage.

Ihnen graut davor, für vielleicht 400 Benutzer und Benutzerinnen eine INI-Datei, welche doch ein etwas unpraktisches Format aufweist, zu führen? Kein Problem, später in diesem Skript wird besprochen, wie Sie die Standardwerte für Ihre Benutzer in Excel administrieren können. Auch eine Excel-Routine, welche es erlaubt auf Knopfdruck aus einer Excel-Tabelle eine INI-Datei zu generieren finden Sie weiter unten, zusammenfassend für alle Methoden, welche in diesem Skript besprochen werden.

Textfelder, Optionsfelder und Kontrollkästchen auf einer *UserForm* mit benutzerspezifischen Standardwerten belegen

▲ Zurück zum Anfang

Um das System zu verstehen machen Sie bitte folgendes Experiment:

Die UserForm

1. Erstellen Sie eine neues Dokument.
2. Wechseln sie nun mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
3. Im VBA-Editor wählen Sie nun vom Hauptmenü den Befehl ***Einfügen - UserForm***.
4. Die UserForm soll für unseres Beispiel den Name **UserForm1** erhalten, Dieser Name ist übrigens der Standardwert, wenn Sie mit einem neuen Dokument beginnen, welches sonst noch keine andere UserForm enthält.
5. Zeichnen Sie nun auf der UserForm mit der Werkzeugsammlung **drei Textfelder**.
6. Fügen Sie der UserForm nun ein **Kontrollkästchen** bei.
7. Erstellen Sie nun auf der selben UserForm einen **Rahmen**, welcher zwei Optionsfelder aufnehmen kann.
8. Platzieren Sie in diesem Rahmen zwei **Optionsfelder**.
9. Wenn Sie so vorgegangen sind und weiter nicht eingegriffen haben beinhaltet Ihre UserForm nun im wesentlichen folgende Elemente:
   - Drei Textfelder mit den Namen **TextBox1**, **TextBox2** und **Texbox3**.
   - Ein Kontrollkästchen mit dem Namen **CheckBox1**.
   - Zwei Optionsfelder mit den Namen **OptionButton1** und **OptionButton2**.
10. Führen Sie nun einen Rechtsklick auf die UserForm aus und wählen sie im Kontextmenü ***Code anzeigen***.
11. Löschen Sie die zwei Code-Zeilen, welche von Word automatisch generiert wurden.
12. Kopieren Sie nun den unten abgedruckten Code in das nun leere Fenster.
13. Speichern Sie nun das Dokument.
14. Rufen Sie nun das kleine Programm **Notepad.exe** (Start - Programme - Zubehör) auf und erstellen Sie damit eine Datei mit den paar wenigen Zeilen, welche unten abgedruckt sind.
15. Speichern Sie diese Datei unter dem Pfad/Namen: **C:\Eigene Dateien\Benutzer.ini**. Sie dürfen der Datei auch einen anderen Namen vergeben oder die Datei in einem anderen Pfad abspeichern, müssten aber in diesem Falle den Wert für die Konstante **Pfad** im Kopf des VBA-Codes noch anpassen.

VBA-Code für die UserForm

```
Private Const Pfad = "C:\Eigene Dateien\Benutzer.ini"

Private Sub UserForm_Activate()
  For Each x In Me.Controls
    If TypeOf x Is MSForms.OptionButton Or _
       TypeOf x Is MSForms.CheckBox Or _
       TypeOf x Is MSForms.TextBox Then
          tmp = ZUE(System.PrivateProfileString(FileName:=Pfad, _
             Section:=Me.Name, Key:=x.Name))
          If TypeOf x Is MSForms.TextBox Then
             x.Text = tmp
          Else
             If IsNumeric(tmp) Then x.Value = Int(tmp)
          End If
    End If
  Next
End Sub

Private Function ZUE(strT As String) As String
  ofs = InStr(strT, "~")
  While ofs > 0
    strT = Left(strT, ofs - 1) & vbCrLf & Right(strT, Len(strT) - ofs)
    ofs = InStr(strT, "~")
  Wend
  ZUE = strT
End Function
```

(Die Werte, welche in der INI-Datei gespeichert sind, werden beim Aufruf der UserForm automatisch in die Kontrollkästchen, so wie auch in die Text- und Optionsfelder der UserForm eingetragen.)

Bemerkungen zum Code

- Beachten Sie, dass es sich beim Prozess **UserForm_Activate** um eine Ereignisroutine handelt. Dieser Name für den Prozess ist also zwingend.
- Für eine produktive Arbeit, sollten Sie den Schaltflächen aussagekräftigere Namen vergeben, als *TextBox1*, *CheckBox1*, etc. Berücksichtigen Sie diese Namensgebung auch in der INI-Datei. Diese müssen selbstverständlich mit den Namen der Schaltflächen übereinstimmen.
- Sie können eine (fast) beliebige Anzahl von Schaltflächen in dieser Art und Weise abhandeln und zwar ohne, dass der Code angepasst werden muss. Vital ist nur, dass die Schaltflächennamen mit den Parametern in der INI-Datei übereinstimmen und zwar buchstäblich und exakt.
- Sie können problemlos in der selben INI-Datei Standardwerte für Schaltflächen in verschiedenen UserForms aufnehmen, dabei dürfen sich die Schaltflächennamen in verschiedenen UserForms durchaus wiederholen. Wählen Sie als Abschnittsname für eine weitere UserForm einfach z.B.**[UserForm2}** (an Stelle von [UserForm1]), kopieren Sie den oben abgedruckten Code auch in diese UserForm. Beachten Sie. dass der Abschnittsname mit dem tatsächlichen Namen der UserForm übereinstimmen muss. Sie verwenden wohl besser einen aussagekräftigeren Namen für die UserForm. Falls Sie UserForms von mehreren Dokumenten oder Dokumentvorlagen wie hier beschrieben abhandeln möchten, denken Sie daran, dass sich die Namen der UserForms in anderen Dokumenten und Dokumentvorlagen wiederholen können. Entweder Sie achten persönlich darauf, dass eine solche Wiederholung nicht geschieht in dem Sie die Namen der UserForms eindeutige vergeben, oder Sie richten für jedes Dokument bzw. für jede Dokumentvorlage eine eigene INI-Datei ein. Dies sind dann schon alle nötigen Vorkehrungen, welche Sie in diesem Falle treffen müssen.

Die INI-Datei

Die dazu passende INI-Datei ist sehr einfach aufgebaut und sieht wie folgt aus:

```
[UserForm1]
TextBox1=Bond J. Mr.
TextBox2=BA007
TextBox3=20000525
CheckBox1= -1
OptionButton1=0
OptionButton2=-1
```

Bemerkungen zur INI-Datei

- Die Reihenfolge der Parameter ist nebensächlich.
- Der Wert in der INI-Datei für ein aktiviertes Kontrollkästchen oder Optionsfeld lautet **-1** , also minus 1. Der Wert für ein deaktiviertes Kontrollkästchen oder Optionsfeld lautet **0**, also null.
- Sollte Ihre UserForm über Textfelder verfügen, welche Zeilenumbrüche zulassen, verwenden Sie bei der Definition der Standardwerte ein Tilde-Zeichen (~) um einen Zeilenumbruch zu signalisieren. Das Definieren des Standardtextes für ein dreizeiliges Textfeld könnte also etwa wie folgt aussehen: `Adresse=Felix Muster~Eulerweg 27~4058 Basel`

Bemerkungen zu diesem Verfahren

- Die oben präsentierte Lösung geht davon aus, dass Sie für jeden Benutzer in Ihrer Organisation eine eigene INI-Datei einrichten. einrichten lassen, bzw. vertreiben. Es mag für Sie mehr Sinn machen, die Standardwerte für alle Benutzer in einer einzigen INI-Datei zu administrieren. Diese Variante wird etwas später in diesem Dokument besprochen.

☛ Ihnen graut davor, für vielleicht 400 Benutzer und Benutzerinnen eine INI-Datei, welche doch ein etwas unpraktisches Format aufweist, zu führen? Kein Problem, später in diesem Skript wird besprochen, wie Sie Die Standardwerte für Ihre Benutzer in Excel administrieren können. Auch eine Excel-Routine, welche es Ihnen erlaubt, auf Knopfdruck aus einer Excel-Tabelle eine INI-Datei zu generieren finden Sie weiter unten.

Kombinationsfeld (*ComboBox*) aus einer INI-Datei laden

▲ Zurück zum Anfang

Im folgenden erfahren Sie, wie ein **Kombinationsfeld** ex INI-Datei geladen werden kann.

Die INI-Datei

Um die Sache zu testen erstellen Sie bitte eine Datei, welchen folgenden Inhalt aufweist:

```
[Elemente]
L1=Wasserstoff
L2=Helium
L3=Lithium
L4=Beryllium
L5=Bor
L6=Kohlenstoff
L7=Stickstoff
L8=Sauerstoff
L9=Flor
L10=Neon
L11=Natrium
L12=Magnesium
```

Der Code um das Kombinationsfeld mit dem Namen **ComboBox1** zu laden lautet nun wie folgt: Um das Verfahren zu testen, kopieren Sie den oben abgebildeten Code in eine UserForm, welche mit einem Kombinationsfeld mit dem Namen **Combobox1** bestückt ist.

VBA-Code

```
Private Const INIDatei = "C:\Elemente.ini"

Private Sub UserForm_Initialize()
  chk = Dir(INIDatei)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & INIDatei & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    Exit Sub
  End If
  i = 1
  x = KombinationsfeldEintragLaden("L" & i)
  While Not x = ""
    Me.ComboBox1.AddItem x
    i = i + 1
    x = KombinationsfeldEintragLaden("L" & i)
  Wend
  Me.ComboBox1.ListIndex = 0
End Sub

Private Function KombinationsfeldEintragLaden(Key As String) As String
  KombinationsfeldEintragLaden = _
     System.PrivateProfileString(FileName:=INIDatei, _
        Section:="Elemente", Key:=Key)
End Function
```

(Das Kombinationsfeld mit dem Namen *ComboBox1* wird ex INI-Datei geladen.)

Eine INI-Datei für alle Benutzer und Benutzerinnen in Ihrer Organisation

Sollten Sie es vorziehen, die Standardwerte der Schaltflächen für eine bestimmte UserForm für alle Benutzer und Benutzerinnen in einer einzigen INI-Datei zu führen, dann muss das Programm als erstes erkennen, wer nun am Rechner sitzt. Als Schlüssel dazu eignet sich insbesondere die **Windows NT-**Kennung unter der sich die Person bei Windows anmeldet. Alternativ könnte man auch den Wert heranziehen, welche man in Word im Dialog ***Extras - Optionen - Benutzerinformationen - Initialen:*** oder ***Name:*** vorfindet. Die nachfolgenden Routinen decken jedenfalls diese drei Möglichkeiten ab. Die Methode mit der Windows-Kennung sollte auch unter **Windows 98** funktionieren, falls Sie den Rechner als Mehrplatzrechner einsetzen.

Sie finden nun zuerst ein Beispiel, wie die INI-Datei dazu aussehen sollte. Daran anschliessend den allgemeingültigen Code einerseits um die UserForm aufzurufen und anderseits für die UserForm. selbst. Der Code verwendet ein Windows-API um die Benutzererkennung zu erfragen. Da derartige Routinen in einer UserForm nicht erlaubt sind, wird diese Aufgabe in ein Standardmodul ausgelagert, welches seinerseits dann die UserForm aufruft.

Die INI-Datei

Angenommen, Ihre Organisation umfasst drei Mitarbeiter, welche die NT-Kennung *jwg*, *hwm* und *hjs* besitzen, dann sieht die INI-Datei wie folgt aus:

```
[jwg]
TextBox1=Goethe W.. Mr.
TextBox2=LH123
TextBox3=20000512
CheckBox1= -1
OptionButton1=0
OptionButton2=-1

[hwm]
TextBox1=Meier H.. Mr.
TextBox2=SR100
TextBox3=20000602
CheckBox1= 0
OptionButton1=0
OptionButton2=-1

[fjs]
TextBox1=Strauss F. Mr.
TextBox2=KL200
TextBox3=20000522
CheckBox1= 0
OptionButton1=-1
OptionButton2=-0
```

VBA-Code zum Aufruf der UserForm

Hier also die Routine, welche die Kennung des Benutzers ermittelt und dann die UserForm aufruft. Stellen Sie sicher, dass die UserForm den Namen **Flugbuchung** trägt, oder passen Sie im Code die Zeile `Flugbuchung.Show` entsprechend an.

```
'*** Beginn des anpassbaren Teiles
Public Const pblPfad = "C:\Eigene Dateien\Benutzer.ini"
Private Const Kennung = "NT-Kennung" 'NT-Kennung; Word-Initialen; Word-Name
'*** Ende des anpassbaren Teiles

Declare Function WNetGetUser Lib "mpr.dll" _
  Alias "WNetGetUserA" (ByVal lpName As String, _
     ByVal lpUserName As String, lpnLength As Long) As Long
Private Const NoError = 0
Public pblSektion As String

Sub FlugbuchungAufruf()
  chk = Dir(pblPfad)
  If chk = "" Then
    MsgBox "Die Initialisierungsdatei mit dem Namen " & Chr(34) & pblPfad & _
       Chr(34) & " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  Select Case Kennung
    Case "Word-Initialen"
      pblSektion = Application.UserInitials
    Case "Word-Name"
      pblSektion = Application.UserName
    Case Else
      pblSektion = NTKennung
  End Select
  Flugbuchung.Show
End Sub

Private Function NTKennung()
  Const lpnLength As Integer = 255
  Dim status As Integer
  Dim lpName, lpUserName As String
  lpUserName = Space$(lpnLength + 1)
  status = WNetGetUser(lpName, lpUserName, lpnLength)
  If status = NoError Then
    lpUserName = Left$(lpUserName, InStr(lpUserName, Chr(0)) - 1)
  End If
  NTKennung = lpUserName
End Function
```

(Ermittelt die Benutzerkennung und ruft dann die UserForm mit dem Namen *Flugbuchung* auf.)

VBA-Code für die UserForm
Hier nun der Code, welche in die UserForm mit dem Namen **Flugbuchung** kopiert werden muss.

```
Private Sub UserForm_Activate()
  For Each x In Me.Controls
    If TypeOf x Is MSForms.OptionButton Or _
       TypeOf x Is MSForms.CheckBox Or _
       TypeOf x Is MSForms.TextBox Then
          tmp = ZUE(System.PrivateProfileString(FileName:=pblPfad, _
             Section:=pblSektion, Key:=x.Name))
          If TypeOf x Is MSForms.TextBox Then
             x.Text = tmp
          Else
             If IsNumeric(tmp) Then x.Value = Int(tmp)
          End If
    End If
  Next
End Sub

Private Function ZUE(strT As String) As String
  ofs = InStr(strT, "~")
  While ofs > 0
    strT = Left(strT, ofs - 1) & vbCrLf & Right(strT, Len(strT) - ofs)
    ofs = InStr(strT, "~")
  Wend
  ZUE = strT
End Function
```

(Der Code für die UserForm mit dem Namen *Flugbuchung*.)

Anmerkungen
Hier nochmals in Kürze, was die Mimik bezwecken will und was von Ihnen erwartet wird:

1. Setzen Sie den Wert der Konstante **Pfad** auf den tatsächlichen Namen Ihrer INI-Datei.
2. Entscheiden Sie, ob der Benutzer auf Grund seiner NT-Kennung, seinen Benutzerinitialen in der Word-Anwendung oder seines Benutzername in der Word-Anwendung erkannt werden soll. Entsprechend Ihrer Wahl setzen Sie den Wert der Konstante **Kennung** auf einen der folgenden möglichen Werte: **NT-Kennung**, **Word-Initialen**, **Word-Name**.
3. Die Abschnitte in der INI-Datei, müssen nun gleich lauten wie die NT-Kennung, die Word-Benutzerinitialen oder der Word-Bemutzername. Für eine volle Abdeckung sollte es also für jeden Benutzer in der Organisation einen Abschnitt im folgenden Sinne geben: **[HMeier]**.
4. Kopieren Sie den VBA-Prozess mit dem Namen **FlugbuchungAufruf** in ein neues Modul eines Dokumentes oder einer Dokumentvorlage.
5. Kopieren Sie den anschliessenden Code für die UserForm in die UserForm mit dem Namen **Flugbuchung**
6. Ihnen graut davor, für vielleicht 400 Benutzer und Benutzerinnen eine INI-Datei, welche doch ein etwas unpraktisches Format aufweist, zu führen? Kein Problem, später in diesem Skript wird besprochen, wie Sie die Standardwerte für Ihre Benutzer in Excel administrieren können. Auch eine Excel-Routine, welche es erlaubt auf Knopfdruck aus einer Excel-Tabelle eine INI-Datei zu generieren finden Sie weiter unten, zusammenfassend für alle Methoden, welche in diesem Skript besprochen werden.

Aus einer Excel-Tabelle eine INI-Datei generieren

▲ Zurück zum Anfang

Angenommen Ihre Organisation zählt z.B. 400 Benutzer und Benutzerinnen und Sie stehen nun vor der Aufgabe, die Standardwerte für all diese Leute zu administrieren, dann wäre dieses Unterfangen sicher ziemlich umständlich, wenn Sie dafür im aktiven Sinne eine INI-Datei führen müssten. Schon viel bequemer lässt sich dies mittels eines Excel-Tabellenblattes oder einer Access-Tabelle machen.

Der nachfolgende VBA-Code exportiert die Daten eines Excel-Tabellenblattes in eine INI-Datei. Diese INI-Datei hat dann genau dieses Format, wie es für die hier präsentierten Code-Beispiele benötigt wird.

Der Code geht von einer einfachen und naheliegenden Struktur der Excel-Tabelle aus. Der Code sieht eine 2-dimensionale Tabelle vor, wobei die **1. Zeile** der Tabelle die Namen der Formularfelder, Textmarkennamen bzw. die Namen der Schaltflächen auf einer UserForm beinhaltet und zwar buchstabengetreu, wie diese entweder im Formular, in der Dokumentvorlage oder auf der UserForm vorkommen.

In der **1. Spalte** der Excel-Tabelle oder besser gesagt in der **Spalte A** sollen dann nacheinander die vorhandenen Benutzerkennungen Ihrer Organisation erscheinen.

Konkret könnte diese Tabelle also wie folgt aussehen, wobei die erste Spalte den Schlüssel darstellt und noch keinen Formularfeldnamen, Textmarkennamen oder Schaltflächennamen darstellt:

	A	B	C	D	E
1	User-ID	Nachname	Vorname	Datum	Flugnummer
2	jwg	Goethe	Wolfgang	22MAY	LH128
3	fjs	Strauss	Josef	30MAY	AF200
4	ham	Meier	Hans	02JUN	SR182
5	etc.				

Diese Excel-Tabelle wird dann mit der folgenden Prozedur auf Knopfdruck (nachdem Sie das Makro mit einem Schaltknopf auf einer Symbolleiste verbunden haben) in eine INI-Datei umgesetzt, welche die folgende Struktur aufweist. Sollte die INI-Datei noch nicht existieren, wird sie erstellt ansonsten überschrieben. Passen Sie bitte im Kopf des Codes Pfad und Name der INI-Datei nach Ihren Wünschen an. Legen Sie die INI-Datei in einem Verzeichnis ab, auf welches alle Benutzer zugreifen können.

```
[jwg]
Nachname=Goethe
Vorname=Wolfgang
Datum=22MAY
Flugnummer=LH128

[fjs]
Nachname=Strauss
Vorname=Josef
Datum=30MAY
Flugnummer=AF200

[ham]
Nachname=Meier
Vorname=Hans
Datum=02JUN
Flugnummer=SR182
```

Und hier nun also der VBA-Code, welches Daten der Excel-Tabelle in eine INI-Datei exportiert. Die Routine kann auch mit Zeilenvorschüben umgehen, welche Sie in einer Excel-Tabelle mit der Tastenkombination **[Alt + Enter]** eingeben, sollte Ihre UserForm Textfelder enthalten, welche mehrzeilig sind.

Passen Sie im Kopf des Codes den Pfad- und Dateiname der INI-Datei noch an Ihre Bedürfnisse an.

```
Private Const Pfad = "C:\Eigene Dateien\User.ini"
Private Const Titel = "User-INI Datei exportieren"

Sub UserIniExportieren()
  On Error GoTo FehlerAnzeigen
  Close
  Open Pfad For Output As #1
  MaxRows = Cells.SpecialCells(xlLastCell).Row
  MaxCols = Cells.SpecialCells(xlLastCell).Column
  For i = 2 To MaxRows
    Print #1, "[" & Cells(i, 1) & "]"
    For j = 2 To MaxCols
      Print #1, Cells(1, j) & "=" & ZUE(Cells(i, j))
    Next j
    If i < MaxRows Then Print #1, ""
  Next i
  Close
FehlerAnzeigen:
  If Err.Number = 0 Then
    Msg = "Export beendet."
    MsgBox Msg, vbOKOnly, Titel
  Else
    Msg = "Fehlercode" & " = " & Err.Number & vbCrLf & Err.Description
    MsgBox Msg, vbCritical, Titel
  End If
End Sub

Private Function ZUE(tStr)
  ZUE = tStr
  ofs = InStr(1, ZUE, Chr(10))
  While ofs > 0
    ZUE = Left(ZUE, ofs - 1) & "~" & Right(ZUE, Len(ZUE) - ofs)
    ofs = InStr(ofs + 1, tStr, Chr(10))
  Wend
End Function
```

(Exportiert ein Excel-Tabellenblatt in das Format einer typischen INI-Datei.)

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Informationen für die Word-Anwendung auch über einen Windows-Neustart hinaus sichern

► Werte von Textfeldern, Kontrollkästchen und Optionsfeldern einer UserForm speichern und beim nächsten Aufruf der UserForm einsetzen?

Microsoft

Deutsch

► WD2000: Erstellen von Kombinationsfelds mit mehr als 25 Elementen

Englisch

► WD2000: How to Create a Combo Box Containing More Than 25 Items

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Defaults.htm

René Probst, Juni 2000

Änderungen

- 26. Januar 2001 - Dropdown-Formularfeld ex INI-Datei laden
- 26. Januar 2001 - Kombinationsfeld ex INI-Datei laden

Werte von Textfeldern, Kontrollkästchen und Optionsfeldern einer UserForm speichern und beim nächsten Aufruf der UserForm einsetzen?

◄ Zurück zum Hauptindex

Aufgabenstellung

> Ich möchte, dass sich Word die Eingaben, welche der Benutzer in einer *UserForm* macht für den nächsten Aufruf merkt. Word sollte diese Werte auch nach beenden von Word und einem Neustart des Rechners behalten und beim nächsten Aufruf der selben *UserForm* automatisch anzeigen.

Lösungsvarianten

MS Word bietet dazu mindestens drei flexible Handhaben. Die Methoden sind nachfolgend aufgeführt.

1. Unter Verwendung der **Windows-Registrierdatenbank**. Diese Methode ist bezüglich Syntax einfach zu handhaben und wird hier eingehend besprochen.
2. Unter Verwendung einer sogenannten **INI-Datei**. Es handelt sich um eine sehr transparente Mimik, welche flexibel einsetzbar ist. Diese Methode wird im folgenden ausführlich dargelegt.
3. Unter Verwendung von **Dokumentenvariablen**. Bei dieser Mimik werden die Werte im Dokument selbst gespeichert. Auch diese Methode wird hier eingehend besprochen.

Charakteristiken (Vor- und Nachteile) der hier präsentierten Lösungen

- Bei der Lösung mit Dokumentenvariablen, sind die nötigen Standardwerte im Word-Dokument bzw. in der Dokumentenvorlage gespeichert. Wenn Sie die Dokumentenvorlage auf einen anderen Rechner portieren, wandern diese Standardwerte mit. Bei der Lösung mit der Registrierdatenbank ist dies nicht der Fall. All zu viele Auswirkungen hat das aber nicht. Die UserForm wird dann halt für das aller erste Mal, ohne Werte angezeigt.
- Dokumentenvariablen haben den Nachteil, dass Sie im Dokument natürlich nur versteckt vorhanden sind. Normalerweise kriegen Sie also deren Inhalte nicht zu Gesicht. Anders bei der Registrierdatenbank. Hier können Sie z.B. im Problemfall mit dem Registriereditor nachschauen, was tatsächlich für Werte abgespeichert sind. Allenfalls lassen sich diese Werte auch dort anpassen. Analoges gilt bei der Lösung mit der INI-Datei
- Der programmtechnische Umgang mit Dokumentenvariablen ist zwar robust (und die hier gezeigten Beispiele nehmen Ihnen diese Arbeit ab), allerdings etwas umständlich. In dieser Beziehung ist das Hantieren mit Schlüsseln und Werte der Registrierdatenbank bzw. einer INI-Datei einfacher, da Word dafür eine ausgezeichnete Schnittstelle kennt.
- Dokumentenvariablen machen ein Dokument etwas komplexer und blähen es etwas auf.
- Word kennt sogar zwei verschiedene Methoden im Zusammenhang mit der Registrierdatenbank. Die hier eingesetzte Methode ist die einfachere und wurde auch deshalb gewählt, weil die Syntax es erlaubt, zusätzlich noch einen Standardwert mitzugeben.
- Bei der Methode mit der Registrierdatenbank in der hier präsentierten Lösung, muss das Programm die Namen der Schaltflächen explizit kennen, bei den anderen beiden Methoden dagegen nicht.

Empfehlung

Meine persönliche Empfehlung geht dahingehend:

Falls Sie eine einfache und schnell zu implementierende Lösung suchen, verwenden Sie die erste Lösung. Wenn Sie eine flexible und schnell ausbaubare Lösung suchen, verwenden Sie die zweite Lösung, wenn Sie partout darauf angewiesen sind, dass die Werte direkt im Dokument bzw. der Dokumentenvorlage gespeichert sind (weil Sie das Dokument vielleicht verschicken müssen), verwenden Sie die dritte Lösung. Die letzte Methode ist für mich die am wenigsten transparente Lösung, sollte mal was schief laufen.

Allgemeine Mimik für alle drei Methoden

Eine UserForm kennt u.a. die zwei Ereignisprozeduren mit den Namen **UserForm_Activate** und **UserForm_Terminate**. Der Code in diesem Skript macht sich diese Ereignisroutinen zu Nutze. Beim *Activate-Ereignis* werden die Werte, falls bereits vorhanden ausgelesen und in die diversen Schaltflächen eingefügt, beim *Terminate-Ereignis* werden die Werte von den Schaltflächen in die Registrierdatenbank. INI-Datei oder in Dokumentenvariablen übertragen.

Der hier präsentierte Code sollte also in all jene UserForms kopiert werden, welche von der Mimik profitieren sollen.

Allgemeine Mimik für alle drei Methoden

Eine UserForm kennt u.a. die zwei Ereignisprozeduren mit den Namen **UserForm_Activate** und **UserForm_Terminate**. Der Code in diesem Skript macht sich diese Ereignisroutinen zu Nutze. Beim *Activate-Ereignis* werden die Werte, falls bereits vorhanden ausgelesen und in die diversen Schaltflächen eingefügt, beim *Terminate-Ereignis* werden die Werte von den Schaltflächen in die Registrierdatenbank. INI-Datei oder in Dokumentenvariablen übertragen.

Der hier präsentierte Code sollte also in all jene UserForms kopiert werden, welche von der Mimik profitieren sollten.

Vorarbeiten für alle drei Methoden

Erstellen Sie eine kleine UserForm, wie es im folgenden Abschnitt Schritt für Schritt erklärt wird.

1. Erstellen Sie eine neues Dokument.
2. Wechseln sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
3. Im VBA-Editor wählen Sie nun vom Hauptmenü den Befehl ***Einfügen - UserForm.***
4. Die UserForm soll für unseres Beispiel den Name **UserForm1** erhalten. Dieser Name ist übrigens der Standardwert, wenn Sie mit einem neuen Dokument beginnen, welches sonst noch keine andere UserForm enthält.
5. Zeichnen Sie nun auf der UserForm mit der Werkzeugsammlung **drei Textfelder**.
6. Fügen Sie der UserForm nun ein **Kontrollkästchen** bei.
7. Erstellen Sie nun auf der selben UserForm eine **Rahmen**, welcher drei Optionsfelder aufnehmen kann.
8. Platzieren Sie in diesem Rahmen drei **Optionsfelder**.
9. Wenn Sie so vorgegangen sind und weiter nicht eingegriffen haben beinhaltet Ihre UserForm nun im wesentlichen folgende Elemente:
   - Drei Textfelder mit den Namen **TextBox1**, **TextBox2** und **Texbox3**
   - Ein Kontrollkästchen mit dem Namen **CheckBox1**.
   - Drei Optionsfelder mit den Namen **OptionButton1**, **OptionButton2** und **OptionButton3**
10. Führen Sie nun einen Rechtsklick auf die UserForm aus und wählen sie im Kontextmenü ***Code anzeigen***.
11. Löschen Sie die zwei Codezeilen, welche von Word automatisch generiert wurden.
12. Kopieren Sie den Code einer der hier besprochenen Variante in das nun leere Fenster.
13. Speichern Sie nun das Dokument.

In Zusammenarbeit mit der Registrierdatenbank

Testen Sie die Mimik doch gleich mal aus. Kopieren Sie den folgenden Code in die UserForm, welche Sie zwischenzeitlich nach der oben aufgeführten Anweisung erstellt haben

Der VBA-Code

```
Private Const Anw = "Flugbuchung"

Private Sub UserForm_Activate()
  S = Me.Name
  TextBox1.Text = GetSetting(AppName:=Anw, Section:=S, _
     Key:="TextBox1", Default:="Name des Fluggastes")
  TextBox2.Text = GetSetting(AppName:=Anw, Section:=S, _
     Key:="TextBox2", Default:="LHxxx")
  TextBox3.Text = GetSetting(AppName:=Anw, Section:=S, _
     Key:="TextBox3", Default:=Format(Date, "ddmm"))
  CheckBox1.Value = Int(GetSetting(AppName:=Anw, Section:=S, _
     Key:="CheckBox1", Default:=0))
  OptionButton1.Value = Int(GetSetting(AppName:=Anw, Section:=S, _
     Key:="OptionButton1", Default:=0))
  OptionButton2.Value = Int(GetSetting(AppName:=Anw, Section:=S, _
     Key:="OptionButton2", Default:=-1))
  OptionButton3.Value = Int(GetSetting(AppName:=Anw, Section:=S, _
     Key:="OptionButton3", Default:=0))
End Sub

Private Sub UserForm_Terminate()
  S = Me.Name
  SaveSetting AppName:=Anw, Section:=S, _
     Key:="TextBox1", Setting:=TextBox1.Text
  SaveSetting AppName:=Anw, Section:=S, _
     Key:="TextBox2", Setting:=TextBox2.Text
  SaveSetting AppName:=Anw, Section:=S, _
     Key:="TextBox3", Setting:=TextBox3.Text
  SaveSetting AppName:=Anw, Section:=S, _
     Key:="CheckBox1", Setting:=Int(CheckBox1.Value)
  SaveSetting AppName:=Anw, Section:=S, _
     Key:="OptionButton1", Setting:=Int(OptionButton1.Value)
  SaveSetting AppName:=Anw, Section:=S, _
     Key:="OptionButton2", Setting:=Int(OptionButton2.Value)
  SaveSetting AppName:=Anw, Section:=S, _
     Key:="OptionButton3", Setting:=Int(OptionButton3.Value)
End Sub
```

(Beim Beenden der UserForm werden die Werte von Textfeldern, Kontrollkästchen und Optionsfelder in die Registrierdatenbank geschrieben und stehen dann beim nächsten Aufruf der UserForm zur Verfügung.

Anmerkungen

- Für das obige Beispiel gilt, die Daten werden im folgenden Schlüssel der Registrierdatenbank gespeichert:
  HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Flugbuchung\UserForm1.
  Der erste Teil des Schlüssels ist von Word so vorgeben, der nächste Teil im Schlüssel ist der Teil, welcher Word als **AppName** bezeichnet und schliesslich folgt noch der Teil, welcher in der Syntax **Section** genannt wird.
- Wie Sie an der Syntax erkennen können, erlaubt Ihnen Word noch das Mitgeben eines Standardwertes. Dieser Wert kommt dann zum tragen, wenn dafür kein Eintrag in der Registrierdatenbank gefunden wird, also in der Regel, wenn die UserForm zum aller ersten Mal aufgerufen wird oder aber erweitert wurde. Verwenden Sie für ein Textfeld eine beliebige Zeichenfolge, für Kontrollkästchen und Optionsfelder lautet der Wert immer entweder **0** bzw. **-1**, wobei **0** für deaktiviert, und **-1** (also minus 1) für aktiviert steht. Beachten Sie dass Sie bei diesen *Boolean-Konstanten* mit der **INT**-Funktion arbeiten müssen.

In Zusammenarbeit mit einer INI-Datei

Die folgende Methode arbeitet mit einer einfach formatierten INI-Datei zusammen. Die Mimik ist so ausgelegt, dass das Programm die Namen der Schaltflächen, welche sich auf der UserForm befindet nicht explizit kennen muss. Dies bedeutet, Sie können den unten aufgeführten Code 1:1 in irgend eine UserForm kopieren. Sie brauchen also nichts anzupassen. Die INI-Datei wird den selben Namen erhalten (allerdings die Dateierweiterung **INI**) wie der Name des Dokumentes bzw. der Dokumentenvorlage, welche(s) die UserForm beherbergt. Um einen Namens-Konflikt zu verhindern, wird Sie auch in im selben Verzeichnis erstellt. Eine Variante davon, nämlich unter Einbezug des Projektnamens, wird etwas weiter unten erklärt.

VBA-Code

```
Private Const INIPfad = "*Vorlage" '*Vorlage, *Projektname oder
                                   'voll qualifizierenden Name
Private INIIntern As String

Private Sub UserForm_Activate()
  INIIntern = INIPfad
  If INIIntern = "*Vorlage" Then
    INIIntern = ActiveDocument.AttachedTemplate.FullName
    INIIntern = Left(INIIntern, Len(INIIntern) - 4) & ".ini"
  End If
  If INIIntern = "*Projekt" Then
    INIIntern = Options.DefaultFilePath(wdProgramPath) & _
       "\" & ActiveDocument.VBProject.Name & ".ini"
  End If
  For Each x In Me.Controls
    If TypeOf x Is MSForms.OptionButton Or _
       TypeOf x Is MSForms.CheckBox Or _
       TypeOf x Is MSForms.TextBox Then
          tmp = ZUE(0, System.PrivateProfileString(FileName:=INIIntern, _
             Section:=Me.Name, Key:=x.Name))
          If TypeOf x Is MSForms.TextBox Then
             x.Text = tmp
          Else
             If IsNumeric(tmp) Then x.Value = Int(tmp)
          End If
```

```
    End If
  Next
End Sub

Private Sub UserForm_Terminate()
  For Each x In Me.Controls
    If TypeOf x Is MSForms.OptionButton Or _
       TypeOf x Is MSForms.CheckBox Or _
       TypeOf x Is MSForms.TextBox Then
       If TypeOf x Is MSForms.TextBox Then
          tmp = ZUE(1, x.Text)
       Else
          tmp = Int(x.Value)
       End If
       System.PrivateProfileString(FileName:=INIIntern, _
          Section:=Me.Name, Key:=x.Name) = tmp
    End If
  Next
End Sub

Private Function ZUE(Richtung As Long, strT As String) As String
  'Verwandelt Yeilenumbrüche in X'FF' und umgekehrt
  If Richtung = 0 Then
    ofs = InStr(strT, Chr(255))
    While ofs > 0
      strT = Left(strT, ofs - 1) & vbCrLf & Right(strT, Len(strT) - ofs)
      ofs = InStr(strT, Chr(255))
    Wend
  Else
    ofs = InStr(strT, Chr(13))
    While ofs > 0
      strT = Left(strT, ofs - 1) & Chr(255) & Right(strT, Len(strT) - ofs - 1)
      ofs = InStr(strT, Chr(13))
    Wend
  End If
  ZUE = strT
End Function
```

(Beim Beenden der UserForm werden die Werte von Textfeldern, Kontrollkästchen und Optionsfelder in eine INI-Datei geschrieben und stehen dann beim nächsten Aufruf der UserForm zur Verfügung. Der Code ist allgemeingültig, braucht also die Namen der Schaltflächen auf der UserForm nicht explizit zu kennen.

Anmerkungen

- Die Mimik kommt auch mit Textfeldern, welche Zeilenumbrüche aufweisen klar. Dazu ist die Funktion mit dem Namen *ZUE* im Code zuständig.
- Bliebe eigentlich nur noch zu klären, wie diese INI-Datei heissen soll und wo sie auf Ihrem Rechner gespeichert werden soll. Bei korrekter Parametrisierung im Code, bieten sich Ihnen folgende Möglichkeiten an:

    1. Die INI-Datei erhält den selben Namen, wie die Dokumentenvorlage, welche die INI-Datei anspricht und wird auch im Pfad der Dokumentenvorlage gespeichert, mit der Ausnahme, dass die Dateierweiterung INI statt DOT lautet. Sollte Ihnen dies genehm sein, dann setzen bzw. belassen Sie den Wert der Konstanten **INIDatei** im Code auf den symbolischen Wert ***Vorlage**.
    2. Optional können Sie den Namen der INI-Datei auch frei wählen. In diesem Falle sollten Sie im Kopf des Codes den Wert der Konstante mit dem Namen **INIDatei** auf den tatsächlichen Namen der gewünschten INI-Datei setzen. Die INI-Datei muss voll qualifiziert werden, also mit Pfadangabe, Name und *.ini*, als Dateierweiterung. Sollte diese Datei noch nicht existieren, wird Sie von Word erstellt.
    3. Als letzte Option können Sie den Code auch so einrichten, dass der Name der INI-Datei gleich lautet wie der Projektname Ihrer Arbeit. Dafür müssen Sie aber Ihrem Projekt zuerst einen eindeutigen Namen vergeben. Wie Sie dies bewerkstelligen, wird gleich anschliessend ausführlich beschrieben. Sollten Sie sich für diese Variante entschliessen, dann setzen Sie im Kopf des Codes den Wert der Konstante mit dem Namen **INIDatei** auf den symbolischen Wert ***Projekt**. Damit ist sicher gestellt, dass diese INI-Datei im Office-Programme-Ordner erstellt wird und der Name der Datei gleichlautend dem Projektes ist. (zuzüglich der Dateierweiterung INI.)

Ihrer Arbeit einen eindeutigen Projektnamen vergeben

Diesen Abschnitt müssen Sie nur dann beachten, wenn Sie den oben aufgeführten Code dahingehen verändert haben, dass der Wert der Konstanten **INIDatei** nun ***Projekt** lautet.

Optional erlaubt VBA auch die Vergabe eines Projektnamens. Dieser sollte eindeutig sein, also kein zweites Mal vorkommen. Für die Eindeutigkeit der Projektnamens müssen Sie selbst Sorge tragen. Gehen Sie wie folgt vor:

1. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
2. Lassen Sie sich mit der Tastenkombination **[Strg + R]** den Projekt-Explorer anzeigen.
3. Erweitern Sie die Ansicht im Projekt-Explorer
4. Führen Sie einen Rechtsklick auf das Projekt aus und wählen Sie im Kontextmenü den Befehl ***Eigenschaft von Projekt...***.
5. Im Textfeld mit der Bezeichnung **Projekt** können Sie nun den gewünschten Projektname vergeben.
6. Speichern Sie Ihre Arbeit.

In Zusammenarbeit mit Dokumentenvariablen

Die beiden Mimiken, welche bisher besprochen wurden zeichnen sich dadurch aus, dass die Standardwerte extern, sei es nun in der Registrierdatenbank oder in einer INI-Datei, abgelegt werden. Hier nun noch eine Lösung, beruhend auf Dokumentenvariablen. Diese Variablen sind in der Dokumentenvorlage selbst gespeichert, was dazu führt, dass die Standardwerte automatisch mitwandern, sollten Sie die Vorlage auf einen anderen Rechner portieren.

Testen Sie die Mimik doch gleich mal aus. Kopieren Sie den folgenden Code in die UserForm, welche Sie zwischenzeitlich nach der oben aufgeführten Anweisung erstellt haben.

Zuerst folgt hier nun die Lösung, bei welcher die Dokumentenvariablen in der Vorlage abgespeichert werden, auf der das Dokument basiert, gleich anschliessend, die Mimik, welche die Dokumentenvariablen im Dokument selbst führt.

VBA-Code

```
Private Vorlage As Document
Private var As Variable

Private Sub UserForm_Initialize()
  Set Vorlage = ActiveDocument.AttachedTemplate.OpenAsDocument
  Vorlage.Windows(1).Visible = False
  For Each x In Me.Controls
    varname = Me.Name & "." & x.Name
    If TypeOf x Is MSForms.TextBox Then
      If CheckExistence(varname) Then
        x.Text = Vorlage.Variables(varname).Value
      End If
    End If
    If TypeOf x Is MSForms.CheckBox Or TypeOf x Is MSForms.OptionButton Then
      If CheckExistence(varname) Then
        x.Value = Int(Vorlage.Variables(varname).Value)
      End If
    End If
  Next x
  Vorlage.Close SaveChanges:=False
End Sub

Private Sub UserForm_Terminate()
  Dim oDoc As Document
  Set oDoc = ActiveDocument
  Set Vorlage = ActiveDocument.AttachedTemplate.OpenAsDocument
  Vorlage.Windows(1).Visible = False
  For Each x In Me.Controls
    varname = Me.Name & "." & x.Name
    If TypeOf x Is MSForms.TextBox Then WertSichern varname, x.Text
    If TypeOf x Is MSForms.CheckBox Or TypeOf x Is MSForms.OptionButton Then
      WertSichern varname, Int(x.Value)
    End If
  Next x
  Vorlage.Close SaveChanges:=True
  oDoc.Activate
End Sub
```

```
Private Function WertSichern(ByVal Name As String, Wert As String)
  Flag = CheckExistence(Name)
  If Flag = True And Wert = "" Then
    var.Delete
    Exit Function
  End If
  If Wert = "" Then Exit Function
  If Flag = True Then
    var.Value = Wert
  Else
    Vorlage.Variables.Add Name:=Name, Value:=Wert
  End If
End Function

Private Function CheckExistence(ByVal Name As String) As Boolean
  CheckExistence = False
  For Each var In Vorlage.Variables
    If var.Name = Name Then
      CheckExistence = True
      Exit For
    End If
  Next
End Function
```

(Beim Beenden der UserForm werden die Werte von Textfeldern, Kontrollkästchen und Optionsfelder in Dokumentenvariablen abgelegt und stehen dann beim nächsten Aufruf der UserForm zur Verfügung. Der Code ist allgemeingültig, braucht also die Namen der Schaltflächen auf der UserForm nicht explizit zu kennen. Bei dieser Version, werden die Dokumentenvariablen in der Dokumentvorlage geführt)

Wie versprochen, hier die Alternative. Die Dokumentenvariablen werden beim folgenden Beispiel nicht in der Dokumentvorlage, sondern im Dokument direkt geführt:

```
Private oDoc As Document
Private var As Variable

Private Sub UserForm_Activate()
  Set oDoc = ActiveDocument
  For Each x In Me.Controls
    If TypeOf x Is MSForms.TextBox Then
      If CheckExistence(x.Name) Then
        x.Text = oDoc.Variables(x.Name).Value
      End If
    End If
    If TypeOf x Is MSForms.CheckBox Or TypeOf x Is MSForms.OptionButton Then
      If CheckExistence(x.Name) Then
        x.Value = Int(oDoc.Variables(x.Name).Value)
      End If
    End If
  Next x
End Sub

Private Sub UserForm_Terminate()
```

```
  For Each x In Me.Controls
    If TypeOf x Is MSForms.TextBox Then WertSichern x.Name, x.Text
    If TypeOf x Is MSForms.CheckBox Or TypeOf x Is MSForms.OptionButton Then
      WertSichern x.Name, Int(x.Value)
    End If
  Next x
  oDoc.Save
End Sub

Private Function WertSichern(Name As String, Wert As String)
  Flag = CheckExistence(Name)
  If Flag = True And Wert = "" Then
    var.Delete
    Exit Function
  End If
  If Wert = "" Then Exit Function
  If Flag = True Then
    var.Value = Wert
  Else
    oDoc.Variables.Add Name:=Name, Value:=Wert
  End If
End Function

Private Function CheckExistence(Name As String) As Boolean
  CheckExistence = False
  For Each var In oDoc.Variables
    If var.Name = Name Then
      CheckExistence = True
      Exit For
    End If
  Next
End Function
```

(Beim Beenden der UserForm werden die Werte von Textfeldern, Kontrollkästchen und Optionsfelder in Dokumentvariablen abgelegt und stehen dann beim nächsten Aufruf der UserForm zur Verfügung. Der Code ist allgemeingültig, braucht also die Namen der Schaltflächen auf der UserForm nicht explizit zu kennen. Bei dieser Version, werden die Dokumentvariablen im Dokument selbst geführt)

Links zu diesem Thema

Eigene Links

► Informationen für die Word-Anwendung auch über einen Windows-Neustart hinaus sichern
► Benutzerspezifische Standardwerte für konventionelle Formularfelder, Textmarkentext und UserForm-Controls

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/UFWerte.htm

René Probst, Juni 2000

Änderungen
- --- nil ---

Wie kann ich Schaltflächen einer UserForm dynamisch, während der Laufzeit erstellen?

◄ Zurück zum Hauptindex

Diese Seite

▼ Ausgangslage
▼ Mögliche Einsatzgebiete und Vorteile von dynamisch erstellten UserForms
▼ Dynamische Schaltflächen ohne Ereignisroutinen erstellen
▼ Tabelle der ProgID's (Klassennamen der Schaltflächen)
▼ Dynamische Schaltflächen mit Ereignisroutinen erstellen
▼ Dynamische einen Rahmen, welcher Schaltflächen beinhaltet erstellen
▼ Textmarkentext über eine UserForm eingeben
▼ Tabelle auf einer UserForm abbilden
▼ UserForm Schaltflächen für ein Flugbuchungsformular dynamisch generieren
▼ UserForm gänzlich dynamisch generieren
▼ VBA und Generische Ereignisprozeduren
▼ Grenzen einer UserForm

Ausgangslage

▲ Zurück zum Anfang

> 1. Ich sollte eine UserForm mit Schaltflächen bestücken. Während der Entwurfszeit weiss ich aber noch nicht, wie viel Schaltflächen letztlich benötigt werden. Dies hängt von den Umständen, bzw. von der Anzahl der Datensätze ab, welche meinem Programm übergeben werden. Wie kann ich alle oder ein Teil der Schaltflächen während der Laufzeit dynamisch erstellen?
>
> 2. Auf einer UserForm erstelle ich dynamische Schaltflächen. Es gelingt mir aber nicht, eine Ereignisprozedur für diese Schaltflächen einzubringen bzw. sie springt nicht an.

Antwort

Folgendes Skript soll diese Fragen beantworten. Als Zückerchen zum Schluss, wird noch der VBA-Code vorgestellt, welcher es ermöglicht, eine UserForm ganz (also inklusive der UserForm selbst) zur Laufzeit zu erstellen.

Mögliche Einsatzgebiete und Vorteile von dynamisch erstellten UserForms

▲ Zurück zum Anfang

- Sie wissen zur Entwurfszeit z.B. nicht, wie viele Textfelder Sie dereinst benötigen, um alle Felder oder Daten abzubilden.
- Im Gegensatz zu reinem VBA-Code können Sie eine Anwendung, welche auf einer UserForm beruht nicht in Textform übermitteln. Sie könnten die Arbeit als Bestandteil eines Word-Dokumentes übermitteln oder Sie könnten die UserForm exportieren, wobei zwei Dateien entstünden, welche der Empfänger dann wieder importieren müsste. Als Alternative zu diesen beiden Methoden, könnten Sie alle Schaltflächen dynamisch erstellen. Der Empfänger müsste dann nur noch eine UserForm einfügen und in dessen Code-Fenster Ihr übermittelter Code.
- In speziellen Fällen, ist das Erstellen von Schaltflächen mittels Code schneller, als wenn Sie die Schaltflächen zeichnen und präzise von Hand ausrichten müssten. Vergleichen Sie dazu, die Routine, welche eine Word-Tabelle auf einer UserForm abbildet.

Dynamische Schaltflächen ohne Ereignisroutinen erstellen

▲ Zurück zum Anfang

Im folgenden eine kleine Übung zum Aufwärmen. Das Beispiel geht von einer leeren UserForm aus und erstellt dynamisch ein Bezeichnungsfeld mit dem Text "Hallo Welt".

Kopieren Sie den abgedruckten Code in eine neue UserForm und testen Sie die Sache.

```
Private Sub UserForm_Initialize()
  Dim lbl1 As MSForms.Label
  Set lbl1 = Me.Controls.Add("Forms.Label.1", , True)
  With lbl1
    .Top = 6
    .Left = 9
    .Width = 90
    .Height = 18
    .Caption = "Hallo Welt"
  End With
End Sub
```

(Erstellt - dynamisch - ein Bezeichnungsfeld auf einer UserForm.)

Anmerkungen zum Code

- Um die Schaltflächen dynamisch einzubringen, mache man sich die Ereignis-Routine mit dem reservierten Namen **UserForm_Initialize** zu Nutze.
- Die Syntax um eine Schaltfläche auf einer UserForm hinzuzufügen besteht aus folgenden drei Teilen:
  - Der erste Parameter legt fest, um welche Art von Schaltfläche es sich handeln soll. Eine Auflistung dieser Begriffe (auch *ProgID* genannt) finden Sie gleich anschliessend.
  - Der zweite Parameter legt den Namen der Schaltfläche fest. Dieser Parameter ist optional. Bei einfachen Übungen, können Sie diesen Parameter wie im Beispiel weglassen. Office vergibt dann automatisch einen Namen nach dem Muster *Label1*, *Label2* etc.
  - Der dritte Parameter gibt an, ob die Schaltfläche sichtbar sein soll. In den meisten Fällen sollten Sie hier den Wert **True** setzen.
- Damit die Schaltfläche überhaupt angezeigt werden kann, müssen Sie noch zusätzliche Eigenschaften für die Schaltfläche setzen, nämlich im Minimum folgende:
  - Obere Position (**Top**) in Pixel bzw. Punkte.
  - Linke Position (**Left**) in Pixel bzw. Punkte.
  - Breite der Schaltfläche (**Width**) in Pixel bzw. Punkte.
  - Höhe der Schaltfläche (**Height**) in Pixel bzw. Punkte.
- Ein Pixel entspricht übrigens dem 72. Teil eines Zolls, das sind dann ca. 0.035 mm.
- Verwenden Sie bei der Deklaration der Schaltfläche `Dim x as MSForms.Label` und nicht etwa die Kurzform `Dim x as Label`, da die Kurzform in einigen Fällen zu schlimmen Verwechslungen innerhalb des Objektmodells führen kann (z.B. CheckBox).

Tabelle der ProgID's (Klassennamen der Schaltflächen)

▲ Zurück zum Anfang

Schaltfläche	ProgID (Klassenname)	Schaltfläche	ProgID (Klassenname)
Anzeige	Forms.Image.1	Listenfeld	Forms.ListBox.1
Befehlsschaltfläche	Forms.CommandButton.1	Multiseiten	Forms.MultiPage.1
Bezeichnungsfeld	Forms.Label.1	Optionsfeld	Forms.OptionButton.1
Bildlaufleiste	Forms.ScrollBar.1	Rahmen	Forms.Frame.1
Drehfeld	Forms.SpinButton.1	Register	Forms.TabStrip.1
Kombinationsfeld	Forms.ComboBox.1	Textfeld	Forms.TextBox.1
Kontrollkästchen	Forms.CheckBox.1	Umschaltfeld	Forms.ToggleButton.1

Dynamische Schaltflächen mit Ereignisroutinen erstellen

▲ Zurück zum Anfang

Das folgende Beispiel geht von einer leeren UserForm aus und fügt dann dieser UserForm ein Bezeichnungsfeld und zwei Befehlsschaltflächen bei. Die beiden Befehlsschaltflächen verfügen über eine Ereignisroutine, damit der Code auf die Wahl des Benutzers reagieren kann.

Betätigt der Benutzer die Schaltfläche mit der Beschriftung **Himmel** dann wird *MS Paint* aufgestartet, ist seine Wahl dagegen **Hölle** dann wird der *Windows-Rechner* aufgestartet.

Kopieren Sie den abgedruckten Code in eine neue UserForm und testen Sie die Sache.

```
Private WithEvents cmd1 As MSForms.CommandButton
Private WithEvents cmd2 As MSForms.CommandButton

Private Sub UserForm_Initialize()
  Me.Height = 80
  Me.Width = 120
  Me.Caption = "Himmel und Hölle"
  Dim lab1 As MSForms.Label
  Set lab1 = Me.Controls.Add("Forms.Label.1", , True)
  With lab1
    .Left = 9
    .Top = 9
    .Height = 12
    .Width = 90
    .Caption = "Treffen Sie Ihre Wahl:"
  End With
  oben = 30
  höhe = 18
  breite = 40
  Set cmd1 = Me.Controls.Add("Forms.CommandButton.1", "cmd1", True)
  With cmd1
    .Left = 9
    .Top = oben
    .Height = höhe
    .Width = breite
    .Caption = "Himmel"
  End With
  Set cmd2 = Me.Controls.Add("Forms.CommandButton.1", "cmd2", True)
  With cmd2
    .Left = 60
    .Top = oben
    .Height = höhe
    .Width = breite
    .Caption = "Hölle"
  End With
End Sub

Private Sub cmd1_Click()
  Shell "pbrush.exe", 1
End Sub

Private Sub cmd2_Click()
  Shell "calc.exe", 1
End Sub
```

(Erstellt - dynamisch - zwei Befehlsschaltflächen, welche über eine Ereignisroutine verfügen, auf einer UserForm.)

Anmerkungen zum Code

- Das Beispiel geht davon aus, dass sich die beiden Programme mit den Namen **calc.exe** und **pbrush.exe** im Windows-Verzeichnis befinden.
- Damit eine Ereignisprozedur für eine Schaltfläche (welche ja zur Entwurfszeit gar noch nicht existiert) geschrieben werden kann, ist es unerlässlich, die Schaltfläche, welche zur Laufzeit hinzugefügt wird im Kopf des Codes zu deklarieren. Diese Deklaration muss zwingend den Parameter **WithEvents** beinhalten. Das ist eigentlich schon der ganze Trick dabei. Sollten Sie in der Vergangenheit glücklos versucht haben eine Ereignisprozedur für eine dynamisch erstellte Schaltfläche zu schreiben, dann haben Sie bestimmt diesen Punkt verfehlt.

! Bitte beachten Sie, dass Ihnen für eine Schaltfläche, welche Sie zur Laufzeit einer UserForm hinzufügen nicht alle Ereignisse zur Verfügung stehen, welche für diese Schaltfläche dokumentiert sind.. Lesen Sie dazu bitte das Kapitel ***Grenzen einer UserForm*** am Ende dieses Skripts.

Dynamische einen Rahmen, welcher Schaltflächen beinhaltet erstellen

▲ Zurück zum Anfang

Das folgende Beispiel geht von einer leeren UserForm aus und fügt dann dieser UserForm zuerst einen Rahmen bei. Auf diesen Rahmen werden dann drei Optionsfelder (*Radio-Buttons*) platziert.

Kopieren Sie den abgedruckten Code in eine neue UserForm und testen Sie die Sache.

```
Private Sub UserForm_Initialize()
  Me.Height = 120
  Me.Width = 150
  Me.Caption = "Radio Buttons"
  Dim Klasse As MSForms.Frame
  Set Klasse = Me.Controls.Add("Forms.Frame.1", , True)
  With Klasse
    .Left = 9
    .Top = 9
    .Height = 60
    .Width = 108
    .Caption = "Wie möchten Sie reisen?"
  End With
  höhe = 12
  breite = 96
  oben = 6
  Dim Radio As MSForms.OptionButton
  For i = 1 To 3
    Set Radio = Klasse.Controls.Add("Forms.OptionButton.1", , True)
    With Radio
      .Name = "Klasse" & i
      .Left = 6
      .Top = oben
      .Width = breite
      .Height = höhe
      Select Case i
        Case 1
          .Caption = "First"
        Case 2
          .Caption = "Buisiness"
        Case 3
          .Caption = "Economy"
          .Value = True
      End Select
    End With
    oben = oben + 15
  Next i
End Sub
```

(Erstellt - dynamisch - ein Rahmen, welcher drei Optionsfelder enthält auf einer UserForm.)

Anmerkungen zum Code

- Bitte beachten Sie, dass das Objekt dem die Optionsfelder hinzugefügt werden nicht die UserForm ist sondern der Rahmen, welcher vorgängig erstellt wurde. Statt `Me.Controls.Add...` müssen Sie also jetzt `MeinRahmen.Controls.Add` codieren.
- Bitte beachten Sie weiter, dass bei der Positionierung der Schaltfläche nicht mehr der Rand der UserForm den Ausgangspunkt hergibt, sondern der Rand des Rahmens. Wenn Sie also z.B. für die Eigenschaft **Top** den Wert *3* angeben, dann klebt das Optionsfeld nicht fast ganz oben auf der UserForm, sondern fast ganz oben am Rahmen.
- Optionsfelder auf dem selben Rahmen schliessen sich gegenseitig aus. Das ist das übliche und auch gewollte Verhalten von Optionsfeldern. Möchten Sie diese Verhalten nicht, verwenden Sie Kontrollkästchen (*CheckBoxes*) an der Stelle von Optionsfelder.

Textmarkentext über eine UserForm eingeben

▲ Zurück zum Anfang

Folgendes Beispiel untersucht das aktive Dokument nach Textmarken. Diese Textmarken werden dann auf einer UserForm dessen Schaltflächen dynamisch erstellt werden, abgebildet. Der Textmarkentext kann dann auf dieser UserForm editiert und wieder in das Dokument zurückgeschrieben werden.

Sie können in der UserForm über *Radio-Buttons* bestimmen, ob die Textmarken in alphabetischer Reihenfolge oder in der Reihenfolge, wie Sie im Hauptteil des Dokumentes vorkommen, erscheinen sollen. Sollten Sie die nicht-alphabetische Reihenfolge wählen, dann bedenken Sie, dass Textmarken, welche in Kopf- und Fusszeilen, sowie in Textfeldern, Fussnoten oder Endnoten platziert sind, nach den Textmarken, welche sich im Hauptteil des Dokumentes befinden, erscheinen werden.

Word kennt eigentlich verschiedene Arten von Textmarken, nämlich offene und geschlossene. Das hier publizierte Programm geht davon aus, dass Sie mit geschlossenen Textmarken arbeiten, also Textmarken, welche einen Bereich im Dokument darstellen. Um eine geschlossene Textmarke in das Dokument einzufügen, muss vorgängig Text markiert werden. Nähere Angaben über die verschiedenen Arten von Textmarken finden Sie im folgenden Kapitel: Arten von Textmarken und ihr Verhalten.

Implementierung der Lösung

Um die Lösung zu testen, verfahren Sie wie folgt:

1. Erstellen Sie ein neues Dokument, schreiben Sie etwas Text, markieren Sie gewisse Textstellen und fügen Sie diesen Textstellen eine Textmarke bei. Die Namen der Textmarken sind frei wählbar. Die Anzahl der Textmarken können Sie ebenfalls frei bestimmen. In unserem Test wird die Anzahl allerdings auf **12** beschränkt. Sie können diese Grenze aber jederzeit im Code auf einfachste Weise anpassen. Setzen Sie dazu den Wert der Konstante mit dem Namen **AnzTM** im Kopf des Codes auf das gewünschte Maxium. Bedenken Sie dabei, dass es keinen Sinn macht, eine UserForm mit unzähligen Schaltflächen zu überladen. MS Office hat auch eine Begrenzung, der möglichen *Controls* auf einer UserForm. Lesen Sie dazu das Kapitel ***Grenzen einer UserForm*** am Ende dieses Skriptes. Da für jede Textmarke zwei Controls gebraucht werden (Bezeichnungsfeld und Textfeld) dürfte also umgeschlagen auf unsere Arbeit etwa bei 200 Textmarken Schluss sein.
2. Wechseln Sie in den VBA-Editor und fügen Sie eine neue UserForm ein. Diese UserForm bleibt leer, da die Textfelder dynamisch, also mit Code erstellt werden.
3. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt ***Code anzeigen***.
4. Löschen Sie die zwei Code-Zeilen, welche Word automatisch generiert hat und kopieren Sie den unten stehenden Code in das nun leere Fenster. Sie können das Beispiel nun testen.

```
Private Const MaxTM = 12
Private WithEvents cmd1 As MSForms.CommandButton
Private WithEvents cmd2 As MSForms.CommandButton
Private WithEvents opt1 As MSForms.OptionButton
Private WithEvents opt2 As MSForms.OptionButton
Private lab() As MSForms.Label
Private tb() As MSForms.TextBox
Private x() As String
Private y() As String
Private AnzTM As Long
Private oDoc As Document

Private Sub UserForm_Initialize()
  Set oDoc = ActiveDocument
  RealTM = oDoc.Bookmarks.Count
  If RealTM = 0 Then
    MsgBox "Dieses Dokument enthält keine Textmarken."
    End
  End If
  AnzTM = RealTM
  If AnzTM > MaxTM Then AnzTM = 12
  ReDim x(RealTM - 1)
  ReDim y(RealTM - 1, 1)
  For i = 0 To RealTM - 1
    x(i) = oDoc.Bookmarks(i + 1).Name
    y(i, 1) = x(i)
    If oDoc.Bookmarks(i + 1).Range.StoryType = wdMainTextStory Then
      y(i, 0) = Format(oDoc.Bookmarks(i + 1).Range.Start, "00000000")
    Else
      y(i, 0) = 1000000000
    End If
  Next i
  WordBasic.SortArray y()
  Me.Width = 360
  Me.Height = AnzTM * 21 + 48
  Me.Caption = oDoc.BuiltInDocumentProperties("Title").Value
  ReDim lab(AnzTM)
  ReDim tb(AnzTM)
  links = 9
  oben = 6
  höhe = 15
  breiteLab = 108
  breiteTB = 210
  abstand = 6
  For i = 1 To AnzTM
    Set lab(i) = Me.Controls.Add("Forms.Label.1", , True)
    With lab(i)
      .Left = links
      .Top = oben
      .Height = höhe
      .Width = breiteLab
    End With
```

```
  Set tb(i) = Me.Controls.Add("Forms.TextBox.1", "TB" & i, True)
  With tb(i)
    .Left = links + breiteLab + abstand
    .Top = oben
    .Height = höhe
    .Width = breiteTB
  End With
  oben = oben + 18
Next i
Dim CB As MSForms.CommandButton
breite = 60
For i = 1 To 2
  Set CB = Me.Controls.Add("Forms.CommandButton.1", "cmd" & i, True)
  With CB
    .Left = (i - 1) * breite + (i * links)
    .Top = Me.Height - 42
    .Height = 18
    .Width = breite
    Select Case i
      Case 1
        Set cmd1 = CB
        .Caption = "Einsetzen"
        .Accelerator = "E"
        .Default = True
      Case 2
        Set cmd2 = CB
        .Caption = "Schliessen"
        .Accelerator = "S"
        .Cancel = True
    End Select
  End With
Next i
Dim Rahmen As MSForms.Frame
Set Rahmen = Me.Controls.Add("Forms.Frame.1", , True)
With Rahmen
  .Left = 150
  .Top = Me.Height - 51
  .Height = 27
  .Width = 186
  .Caption = "Sortiern nach:"
End With
For i = 1 To 2
  Set OB = Rahmen.Controls.Add("Forms.OptionButton.1", "opt" & i, True)
  With OB
    .Top = 3
    .Height = 15
    Select Case i
      Case 1
        Set opt1 = OB
        .Left = 3
        .Width = 60
        .Caption = "Namen"
      Case 2
```

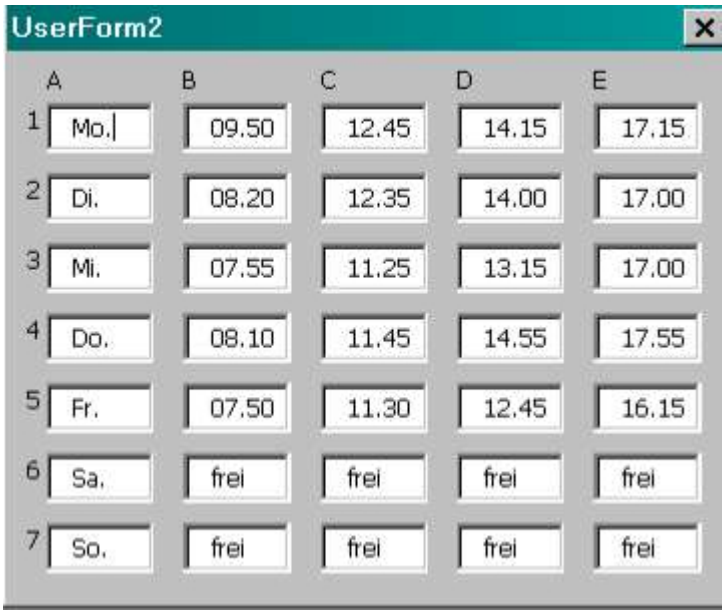
```
          Set opt2 = OB
          .Left = 72
          .Width = 90
          .Caption = "Position im Text"
          .Value = True
      End Select
    End With
  Next i
End Sub

Private Sub cmd1_Click()
  Dim TMRange As Range, TMName As Variant
  For i = 1 To AnzTM
    TMName = Left(lab(i).Caption, Len(lab(i).Caption) - 1)
    Set TMRange = ActiveDocument.Bookmarks(TMName).Range
    TMRange.Text = tb(i).Text
    ActiveDocument.Bookmarks.Add Name:=TMName, Range:=TMRange
  Next i
  Unload Me
End Sub

Private Sub cmd2_Click()
  Unload Me
End Sub

Private Sub opt1_Click()
  TextmarkenLaden 0
End Sub

Private Sub opt2_Click()
  TextmarkenLaden 1
End Sub

Private Function TextmarkenLaden(Sortierung As Long)
  For i = 1 To AnzTM
    Select Case Sortierung
      Case 0
        lab(i).Caption = x(i - 1) & ":"
        tb(i).Text = oDoc.Bookmarks(x(i - 1)).Range.Text
      Case 1
        lab(i).Caption = y(i - 1, 1) & ":"
        tb(i).Text = oDoc.Bookmarks(y(i - 1, 1)).Range.Text
    End Select
  Next i
End Function
```

(Die nötigen Bezeichnungs- und Textfelder um die Textmarken im Dokument auf einer UserForm darzustellen, werden dynamisch erstellt. Das gleiche gilt für die Befehlsschaltflächen, um den Textmarkentext zurück zu schreiben.)

Tabelle auf einer UserForm abbilden

▲ Zurück zum Anfang

Folgendes Beispiel bildet eine Tabelle, welches die erste Tabelle im Word-Dokument darstellt auf einer UserForm ab.



Implementierung der Lösung

Um die Lösung zu testen, verfahren Sie wie folgt:

1. Erstellen Sie ein neues Word-Dokument und fügen sie eine Tabelle ein. Grundsätzlich kann diese Tabelle eine beliebige Anzahl von Zeilen und Spalten aufweisen. Das Programm ermittelt die Grösse der Tabelle dynamisch. Da die UserForm aber nicht beliebig gross werden sollte, ist die Grösse im Code bewusst limitiert (12 Zeilen und 6 Spalten). Fühlen Sie sich aber frei, diese beiden Konstanten im Kopf des Prozesses mit dem Namen **DynamischeSchaltflächenTabelleAufruf** nach Ihren Bedürfnissen nach oben anzupassen. Die Namen der Konstanten lauten **MaxZeilen** bzw. **MaxSpalten**. Bedenken Sie aber dabei, dass die Anzahl der *Controls*, welche eine UserForm beherbergen kann nach oben limitiert ist. Lesen Sie dazu das Kapitel ***Grenzen einer UserForm*** am Ende diese Skriptes.
2. Füllen Sie die Zellen mit etwas Text, damit das Ergebnis später auch sichtbar wird.
3. Wechseln Sie in den VBA-Editor und fügen Sie ein neues Module ein.
4. Kopieren Sie den Prozess mit dem Namen **DynamischeSchaltflächenTabelleAufruf** in dieses Modul. Beachten Sie, dass diese Routine davon aus geht, dass die UserForm den Namen *UserForm1* trägt. Allenfalls muss das noch angepasst werden.
5. Fügen Sie eine neue UserForm ein. Diese UserForm bleibt leer, das die Textfelder dynamisch, also mit Code erstellt werden.
6. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt ***Code anzeigen***.
7. Löschen Sie die zwei Code-Zeile, welche Word automatisch generiert hat und kopieren Sie den unten stehenden Code, welcher für die UserForm bestimmt ist, in das nun leere Fenster..
8. Sie sind nun schon bereit, um die Sache zu testen.

VBA-Code

Aufruf der UserForm

Folgender Code tätigt die nötigen Vorarbeiten und ruft dann die UserForm auf.

```
Private Const MaxZeilen = 12
Private Const MaxSpalten = 6

Sub DynamischeSchaltflächenTabelleAufruf()
  If ActiveDocument.Tables.Count = 0 Then
    MsgBox "Im Dokument befindet sich keine Tabelle."
    End
  End If
  Dim Tabelle As Table
  Set Tabelle = ActiveDocument.Tables(1)
  AnzZeilen = Tabelle.Rows.Count
  If AnzZeilen > MaxZeilen Then AnzZeilen = MaxZeilen
  AnzSpalten = Tabelle.Columns.Count
  If AnzSpalten > MaxSpalten Then AnzSpalten = MaxSpalten
  ReDim x(AnzZeilen - 1, AnzSpalten - 1) As String
  For i = 0 To AnzZeilen - 1
    For j = 0 To AnzSpalten - 1
      tmp = Tabelle.Cell(i + 1, j + 1).Range.Text
      x(i, j) = Left(tmp, Len(tmp) - 2)
    Next j
  Next i
  UserForm1.DynamischeSchaltflächenTabelleUserForm x()
End Sub
```

(Die Tabelle wird in ein zweidimensionales Datenfeld geladen, anschliessend wird die UserForm angezeigt..)

Der Code für die UserForm

Folgender Code erstellt dynamisch die nötigen Textfelder. Kopieren Sie den Code in eine leere User-Form.

```
Private Lab As MSForms.Label
Private TB As MSForms.TextBox

Sub DynamischeSchaltflächenTabelleUserForm(x() As String)
  Dim TB As MSForms.TextBox
  Titel = 24
  Header = 12
  Rand = 6
  StartOben = 3
  StartLinks = 6
  Höhe = 15
  Breite = 32
  AbsX = 9
  AbsY = 6
  Me.Height = ((UBound(x, 1) + 1) * Höhe) + (UBound(x, 1) * AbsY) + _
     (2 * StartOben) + Titel + Header
```

```
  Me.Width = ((UBound(x, 2) + 1) * Breite) + (UBound(x, 2) * AbsX) + _
     (2 * (StartLinks + Rand))
  oben = StartOben
  links = StartLinks + Rand
  For i = 0 To UBound(x, 2)
    Set Lab = Me.Controls.Add("Forms.Label.1", , True)
    With Lab
      .Top = oben
      .Left = links
      .Height = 9
      .Width = Breite
      .Caption = Chr(i + 65)
      links = links + Breite + AbsX
    End With
  Next i
  oben = StartOben + Header
  For i = 0 To UBound(x, 1)
  Set Lab = Me.Controls.Add("Forms.Label.1", , True)
  With Lab
    .Top = oben
    .Left = StartLinks
    .Height = 9
    .Width = 12
    .Caption = i + 1
    .TextAlign = fmTextAlignLeft
    oben = oben + Höhe + AbsY
  End With
  Next i
  oben = StartOben + Header
  For i = 0 To UBound(x, 1)
    links = StartLinks + Rand
    For j = 0 To UBound(x, 2)
      Set TB = Me.Controls.Add("Forms.TextBox.1", "TB" & Chr(j + 64) & i + 1, True)
      With TB
        .Top = oben
        .Left = links
        .Height = Höhe
        .Width = Breite
        .Text = x(i, j)
        links = links + Breite + AbsX
      End With
    Next j
    oben = oben + Höhe + AbsY
  Next i
  Me.Show
End Sub
```

(Die nötigen Textfelder werden dynamisch erstellt. Grösse der Textfelder und die Abstände zwischen den Feldern können Sie im Kopf des Codes allenfalls anpassen.)

UserForm Schaltflächen für ein Flugbuchungsformular dynamisch generieren

▲ Zurück zum Anfang

Folgendes Beispiel geht von einer völlig leeren UserForm aus und erstellt dann dynamisch die Schaltflächen, um eine Flugbuchung zu tätigen. Dabei kommen folgende Elemente zum Einsatz:

- Drei **Textfelder**.
- Zwei **Rahmen** als Gefässe für Optionsfelder und Kontrollkästchen
- Zwei **Kontrollkästchen**
- Drei **Optionsfelder**
- Zwei **Befehlsschaltflächenflächen** zu denen es Ereignisse gibt (Im Beispiel wird das Klick-Ereignis eingesetzt).

Wird das Formular ausgefüllt und dann über die Schaltfläche ***Ok*** geschlossen, wird dem Kunden eine Bestätigung der Flugbuchung ausgedruckt.

Implementierung der Lösung

Um die Lösung zu testen, verfahren Sie wie folgt:

1. Erstellen Sie ein neues Word-Dokument.
2. Wechseln Sie in den VBA-Editor und fügen Sie ein neues Modul ein.
3. Kopieren Sie den Prozess mit dem Namen **DynamischeSchaltflächenFlugbuchungAufruf** in dieses Modul. Beachten Sie, dass diese Routine davon aus geht, dass die UserForm den Namen *frmFlugbuchung* trägt. Allenfalls muss das noch angepasst werden.
4. Fügen Sie eine neue UserForm ein. Diese UserForm bleibt leer, da die Elemente dynamisch, also mit Code erstellt werden.
5. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt ***Code anzeigen***.
6. Löschen Sie die zwei Code-Zeile, welche Word automatisch generiert hat und kopieren Sie den unten stehenden Code, welcher für die UserForm bestimmt ist, in das nun leere Fenster..
7. Sie sind nun schon bereit, um die Sache zu testen.

VBA-Code

Aufruf der UserForm

Folgender Code ruft die UserForm mit dem Namen *frmFlugbuchung* auf.

```
Sub DynamischeSchaltflächenFlugbuchungAufruf()
  frmFlugbuchung.Show
End Sub
```

(Ruft die UserForm auf.)

Code für die UserForm
Folgender Code kopieren Sie bitte in UserForm mit dem Namen *frmFlugbuchung*.

```
Private lbl(2) As MSForms.Label
Private Rahmen(1) As MSForms.Frame
Private txtBox(2) As MSForms.TextBox
Private optFld(2) As MSForms.OptionButton
Private chkBox(1) As MSForms.CheckBox
Private WithEvents cmdOk As MSForms.CommandButton
Private WithEvents cmdCancel As MSForms.CommandButton

Private Sub UserForm_Initialize()
  Me.Caption = "Flugbuchung"
  Me.Height = 144
  Me.Width = 252
  links = 9
  oben = 3
  For i = 0 To 2
    Set lbl(i) = Me.Controls.Add("Forms.Label.1", , True)
    With lbl(i)
      .Left = links
      .Top = oben
      .Height = 12
      .Width = 90
      Select Case i
        Case 0: .Caption = "Name des Passagiers:"
        Case 1: .Caption = "Flugnummer:"
        Case 2: .Caption = "Datum:"
      End Select
    End With
    oben = oben + 30
  Next i
  oben = 15
  For i = 0 To 2
    Set txtBox(i) = Me.Controls.Add("Forms.TextBox.1", , True)
    With txtBox(i)
      .Left = links
      .Top = oben
      .Height = 15
      .Width = 90
    End With
    oben = oben + 30
  Next i
  oben = 10
  For i = 0 To 1
    Set Rahmen(i) = Me.Controls.Add("Forms.Frame.1", , True)
    With Rahmen(i)
      .Left = 120
      .Top = oben
      .Width = 120
      Select Case i
        Case 0:
          .Caption = "Klasse:"
```

```
        .Height = 60
      Case 1:
        .Caption = "Besonderes:"
        .Height = 42
    End Select
  End With
  oben = oben + 66
Next i
oben = 6
For i = 0 To 2
  Set optFld(i) = Rahmen(0).Controls.Add("Forms.OptionButton.1", , True)
  With optFld(i)
    .Left = 6
    .Top = oben
    .Width = 120
    .Height = 12
    Select Case i
      Case 0
        .Caption = "First"
      Case 1
        .Caption = "Buisiness"
      Case 2
        .Caption = "Economy"
        .Value = True
    End Select
  End With
  oben = oben + 15
Next i
oben = 6
For i = 0 To 1
  Set chkBox(i) = Rahmen(1).Controls.Add("Forms.CheckBox.1", , True)
  With chkBox(i)
    .Left = 6
    .Top = oben
    .Width = 120
    .Height = 12
    Select Case i
      Case 0: .Caption = "Rückbestätigt:"
      Case 1: .Caption = "Menber of Qualiflyer:"
    End Select
  End With
  oben = oben + 15
Next i
Dim cmd(1) As MSForms.CommandButton
links = 9
For i = 0 To 1
  Set cmd(i) = Me.Controls.Add("Forms.CommandButton.1", , True)
  With cmd(i)
    .Left = links
    .Top = 99
    .Width = 48
    .Height = 18
    Select Case i
```

```
        Case 0
          .Name = "cmdOk"
          .Caption = "Ok"
          .Accelerator = "O"
        Case 1
          .Name = "cmdCancel"
          .Caption = "Abbrechen"
          .Accelerator = "A"
      End Select
    End With
    links = links + 54
  Next i
  Set cmdOk = cmd(0)
  Set cmdCancel = cmd(1)
End Sub

Private Sub cmdOk_Click()
  For i = 0 To 2
    If optFld(i).Value = True Then
      tmp = optFld(i).Caption
      Exit For
    End If
  Next i
  Text = "Sehr geehrte(r) Frau/Herr " & txtBox(0).Text & Chr(11) _
     & Chr(11) & "Wir haben Sie für den Flug " & txtBox(1).Text & _
        " am " & txtBox(2).Text & " in der " & tmp & " Klasse vorgemerkt."
  Unload Me
  Dim nDoc As Document
  Set nDoc = Documents.Add
  nDoc.Range.InsertAfter Text
  nDoc.PrintOut Background:=False
  nDoc.Close SaveChanges:=False
End Sub

Private Sub cmdCancel_Click()
  Unload Me
End Sub
```

(Erstellt sämtliche Schaltflächen einer UserForm um eine Flugbuchung zu tätigen dynamisch.)

UserForm gänzlich dynamisch generieren

▲ Zurück zum Anfang

Zum Schluss diese Skriptes nun also noch der Code um eine UserForm gänzlich während der Laufzeit zu generieren. Das folgende Code-Beispiel führt zum selben Resultat, wie eines der Beispiele welches hier schon besprochen wurde, nur werden diesmal nicht nur die Schaltflächen auf der UserForm dynamisch generiert, sondern die UserForm ebenfalls.

Implementierung der Lösung

Um die Lösung zu testen, verfahren Sie wie folgt:

1. Erstellen Sie ein neues Word-Dokument.
2. Wechseln Sie in den VBA-Editor und fügen Sie ein neues Modul ein.
3. Kopieren Sie den unten abgedruckten Code in das leere Fenster, welches sich geöffnet hat..
4. Wählen Sie immer noch im VBA-Editor vom Hauptmenü aus den Befehl ***Extras - Verweise...***.
5. Suchen Sie in der nun angezeigten Liste nach dem Eintrag mit dem Namen **Microsoft Visual Basic x.0 Extensibility** und setzen Sie das Häkchen vor diesem Eintrag.
6. Speichern Sie nun das Dokument und rufen Sie das Makro mit dem Namen **UserFormLaufzeitGeneriert** auf.

VBA-Code

```
Sub UserFormLaufzeitGeneriert()
  Q = Chr(34)
  Set Formular = _
     ThisDocument.VBProject.VBComponents.Add(vbext_ct_MSForm)
  With Formular.CodeModule
    .InsertLines 1, "Private lab1 As MSForms.Label"
    .InsertLines 2, "Private WithEvents cmd1 As MSForms.CommandButton"
    .InsertLines 3, "Private WithEvents cmd2 As MSForms.CommandButton"
    .InsertLines 4, "Private Sub UserForm_Initialize()"
    .InsertLines 5, "Me.Height = 80"
    .InsertLines 6, "Me.Width = 120"
    .InsertLines 7, "Me.Caption = " & Q & "Run-Time generated" & Q
    .InsertLines 8, "Set lab1 = Me.Controls.Add(" & Q & "Forms.Label.1" _
       & Q & ", , True)"
    .InsertLines 9, "With lab1"
    .InsertLines 10, ".Left = 9"
    .InsertLines 11, ".Top = 9"
    .InsertLines 12, ".Height = 12"
    .InsertLines 13, ".Width = 90"
    .InsertLines 14, ".Caption = " & Q & "Treffen Sie Ihre Wahl:" & Q
    .InsertLines 15, "End With"
    .InsertLines 16, "Set cmd1 = Me.Controls.Add(" & Q & _
     "Forms.CommandButton.1" & Q & "," & Q & "cmd1" & Q & ", True)"
    .InsertLines 17, "With cmd1"
    .InsertLines 18, ".Left = 9"
    .InsertLines 19, ".Top = 30"
    .InsertLines 20, ".Height = 18"
    .InsertLines 21, ".Width = 40"
    .InsertLines 22, ".Caption = " & Q & "Himmel" & Q
    .InsertLines 23, "End With"
    .InsertLines 24, "Set cmd2 = Me.Controls.Add(" & Q & _
     "Forms.CommandButton.1" & Q & "," & Q & "cmd2" & Q & ", True)"
    .InsertLines 25, "With cmd2"
    .InsertLines 26, ".Left = 60"
    .InsertLines 27, ".Top = 30"
    .InsertLines 28, ".Height = 18"
    .InsertLines 29, ".Width = 40"
    .InsertLines 30, ".Caption = " & Q & "Hölle" & Q
    .InsertLines 31, "End With"
    .InsertLines 32, "End Sub"
    .InsertLines 33, "Private Sub cmd1_Click()"
    .InsertLines 34, "Shell " & Q & "calc.exe" & Q & " ,1"
    .InsertLines 35, "End Sub"
    .InsertLines 36, "Private Sub cmd2_Click()"
    .InsertLines 37, "Shell " & Q & "pbrush.exe" & Q & " ,1"
    .InsertLines 38, "End Sub"
  End With
  Dim Frm As Object
  Set Frm = UserForms.Add(Formular.Name)
  Frm.Show
  ThisDocument.VBProject.VBComponents.Remove Formular
End Sub
```

(Erstellt eine UserForm während der Laufzeit.)

Verweis auf eine Objektbibliothek dynamisch erstellen

Wenn auch der Verweis auf die Objektbibliothek dynamisch erstellt werden soll, dann kopieren Sie zusätzlich zu der Prozedur, welche oben abgedruckt ist folgende beiden Prozeduren in das **selbe** Modul und rufen Sie dann das Makro mit dem Namen **UserFormLaufzeitGeneriertStart** auf.

```
Sub UserFormLaufzeitGeneriertStart()
  ReferenzEinrichten
  Application.OnTime When:=Now + TimeValue("00:00:00"), _
    Name:="UserFormLaufzeitGeneriert"
End Sub

Private Sub ReferenzEinrichten()
  With Application.VBE.ActiveVBProject.References
    On Error Resume Next
    If .Item("VBIDE").IsBroken Then .Remove .Item("VBIDE")
    .AddFromGuid "{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}", 0, 0
    On Error GoTo 0
  End With
End Sub
```

(Zuerst wird der Verweis dynamisch eingerichtet, anschliessend wird der Prozess, welcher die UserForm generiert, a s y n c h r o n aufgerufen.)

VBA und Generische Ereignisprozeduren

▲ Zurück zum Anfang

Sollten Sie *Visual Basic 6.0* oder eine frühere Version von Visual Basic kennen, dann sind sie es gewohnt, dass Sie für eine Gruppe von ähnlich gelagerten Schaltflächen generische Ereignisprozeduren schreiben können. Dabei verwenden Sie einen sogenannten *Array* als Schaltflächennamen in der Form *TextBox(1)*, *TextBox(2)*, *TextBox(3)* etc. und können nun eine Ereignisprozedur schreiben, welche z.B. mit `Private Sub TextBox()` beginnt und damit alle Textfelder mit diesem generischen Namen abdeckt.

VBA in Office bietet dieses Feature leider nicht an. Trotz dieser schlechten Nachricht gibt es aber Möglichkeiten in VBA all zu viel redundanter Code zu vermeiden Dazu werden im folgenden drei Mimiken vorgestellt. Diese Beispiele gehen von folgenden Aufgabenstellungen aus:

Schleife über alle Schaltflächen, welche sich auf der UserForm befinden

Die Aufgabenstellung für das folgende Beispiel ist simpel. Um die Mimik zu testen, füllen Sie eine UserForm mit einigen Textfeldern, welche die Namen **TextBox1**, **TextBox2** und **TextBox3** etc. tragen. Die genau Zahl ist Ihnen freigestellt. Solange Sie sich an das erwähnte Muster bezüglich der Schaltflächennamen halten, muss der Code nicht angepasst werden, wenn zusätzliche Textfelder hinzu kommen. Fügen Sie nun noch eine Befehlsschaltfläche mit dem Namen **CommandButton1** bei. Wenn der Benutzer diese Schaltfläche klickt, soll der Inhalt aller Textfelder, deren Namen mit *TextBox* beginnen, gelöscht werden. Der Code dazu sieht wir folgt aus:

```
Private Sub CommandButton1_Click()
  Dim cntrl As MSForms.Control
  For Each cntrl In Me.Controls
    If Left(cntrl.Name, 7) = "TextBox" Then cntrl.Text = ""
  Next
End Sub
```

(Beginnt der Name der Schaltfläche mit *TextBox* wird deren Inhalt gelöscht.)

Die nächste Aufgabenstellung ist sehr ähnlich wie die Besprochene, nur wird in diesem Beispiel der Text aller Textfelder gelöscht unabhängig vom Namen, welche dies Textfelder tragen. Dafür verwenden wir eine andere Mimik als im letzten Beispiel.

```
Private Sub CommandButton1_Click()
  Dim cntrl As MSForms.Control
  For Each cntrl In Me.Controls
    If TypeOf cntrl Is TextBox Then cntrl.Text = ""
  Next
End Sub
```

(Die Inhalte aller Textfelder auf der UserForm werden gelöscht.)

Code für Ereignisprozeduren Dank Einsatz von Subroutinen auf das Minimum abspecken

Folgende Mimik entbindet uns zwar nicht davor, für jede Schaltfläche eine Ereignisprozedur im Code aufzunehmen, trotzdem kann Dank dem Einsatz von Unterroutinen viel redundanter Code eingespart werden und die Sache wird somit bestimmt übersichtlicher und vor allem auch unterhaltsfreundlicher.

Folgendes Szenario ist angesagt: Eine UserForm enthält zwei Gruppen zu je drei Textfeldern. Die erste Gruppe der Textfelder können vom Benutzer ausgefüllt werden, die Testfelder der zweiten Gruppe dagegen nicht, da dies unser Code erledigen wird. Die drei Eingabefelder sollen so ausgelegt sein, dass der Benutzer eine Zahl (und nur eine Zahl) von 0 bis 65535 (2^{16} - 1) eingeben kann. Falls die Validierung positiv verläuft, wird dann die zweite Gruppe der Textfelder mit den entsprechenden hexadezimalen Werten dieser Zahlen ausgefüllt. Dabei wird die Hexadezimalzahl, welche immer vierstellig angezeigt werden soll gegebenenfalls mit führenden Nullen aufgefüllt.

Um die Sache zu testen, gehen Sie bitte wie folgt vor.

1. Erstellen Sie eine neue UserForm.
2. Platzieren Sie drei Textfelder auf dieser UserForm mit den Namen **Dec1**, **Dec2** und **Dec3**.
3. Platzieren Sie nochmals drei Textfelder auf dieser UserForm diesmal mit den Namen **Hex1**, **Hex2** und **Hex3**
4. Platzieren Sie nun noch eine Befehlsschaltfläche mit den Namen **CommandButton1** auf die UserForm.
5. Kopieren Sie den unten abgedruckten Code in die UserForm und testen Sie die Sache.

```
Private Sub Dec1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
  If Validierung("KeyPress", KeyAscii) Then KeyAscii = 0
End Sub

Private Sub Dec2_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
  If Validierung("KeyPress", KeyAscii) Then KeyAscii = 0
End Sub

Private Sub Dec3_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
  If Validierung("KeyPress", KeyAscii) Then KeyAscii = 0
End Sub

Private Sub Dec1_Exit(ByVal Cancel As MSForms.ReturnBoolean)
  If Validierung("Exit") Then Cancel = True
End Sub

Private Sub Dec2_Exit(ByVal Cancel As MSForms.ReturnBoolean)
  If Validierung("Exit") Then Cancel = True
End Sub

Private Sub Dec3_Exit(ByVal Cancel As MSForms.ReturnBoolean)
  If Validierung("Exit") Then Cancel = True
End Sub

Private Function Validierung(Ereignis As String, _
   Optional ByVal Taste As Long) As Boolean
  Select Case Ereignis
    Case "KeyPress"
      If Taste < 48 Or Taste > 57 Then 'Ziffern 0 bis 9
        Validierung = True
        Beep
      End If
    Case "Exit"
      If Me.ActiveControl.Text > 65535 Then
        Validierung = True
        Beep
        Me.ActiveControl.SelStart = 0
        Me.ActiveControl.SelLength = Len(Me.ActiveControl.Text)
      Else
        tmp = Hex(Me.ActiveControl.Text)
        While Len(tmp) < 4
          tmp = "0" & tmp
        Wend
        corr = Right(Me.ActiveControl.Name, Len(Me.ActiveControl.Name) - 3)
        For Each cntrl In Me.Controls
          If cntrl.Name = "Hex" & corr Then
            cntrl.Value = tmp
            Exit For
          End If
        Next
      End If
  End Select
End Function
```

(Verlagert man die Validierung von Benutzereingaben konsequent in Unterroutinen oder Unterfunktionen, lässt sich das Manko von VBA, nämlich dass Schaltflächen nicht generisch benannt werden können, teilweise ausbügeln.)

Grenzen einer UserForm

▲ Zurück zum Anfang

Beschränkte Zahl von Ereignissen

Jeder Typus von Schaltflächen verfügt über eine bestimmte Gruppe von festgelegten Ereignissen. Was man in der Praxis oft gar nicht realisiert ist die Tatsache, dass diese Ereignisse sich in zwei Kategorien aufteilen lassen. Einerseits gibt es die Untergruppe der Ereignisse, welche der Schaltfläche direkt angehören und andererseits eine Untergruppe von Ereignissen, welche dem Container dieser Schaltfläche. in unserem Falle also der UserForm, angehören.

Bei einer statisch erzeugten Schaltfläche, also einer Schaltfläche, welche Sie explizit auf die UserForm malen, stehen Ihnen beide Untergruppen von Ereignissen zur Verfügung, bei dynamisch erzeugten Schaltflächen leider nur die Untergruppe, welche der Schaltfläche selbst zugehört. Konkret bedeutet dies, dass Sie z.B. für ein Textfeld folgende Ereignisse **nicht** abfangen können:

- AfterUpdate
- BeforeUpdate
- Enter
- Exit

Folgende Ereignisse stehen Ihnen allerdings zur Verfügung:

- BeforeDragOver
- BeforeDropOrPaste
- Change
- DblClick
- DropButtonClick
- Error
- KeyDown
- KeyPress
- KeyUp
- MouseDown
- MouseMove
- MouseUp

Beschränkte Zahl von Schaltflächen

Spätestens, wenn man eine UserForm dynamisch mit Schaltflächen bestückt, sollte eigentlich die Frage auftauchen, ob es diesbezüglich eine Obergrenze gibt. Wie so oft bei Microsoft-Produkten, schweigt sich die Hilfe und die Dokumentation bezüglich Beschränkungen aus. Man stösst halt dann unvermittelt beim Entwickeln darauf und muss sein Projekt redimensionieren oder einen anderen Weg suchen und finden. Nach meiner Erfahrung geht dabei mindestens 30% der Arbeit, welche man geleistet hat gleich wieder flöten.

Nun diese Obergrenze gibt es. Sie liegt wie die Praxis ergeben hat und wie auch ein Artikel auf der *knowledge base* von Microsoft aussagt bei **511 Controls**. Diese Zahl stimmt zu mindest für **Word 97**. Anders als bei qualitativ hochwertigen und professionellen Entwicklungsoberflächen (von denen ich einige kennen lernen durfte) erfolgt, wenn man diese Grenze überschreitet keine spezifische Fehlermeldung. Im *Debug-Modus* erhalten Sie wenigsten noch die Fehlermeldung *Stapelspeicherüberlauf*. Im operationellen Modus ist sich Microsoft zu vornehm um so etwas abzufangen. Die stolze Word-Anwendung schmiert in diesem Falle ganz einfach sang- und klanglos ab.

Tja, dazu fällt mir eigentlich nur etwas ein: *It's Microsoft!* - oder frei ins Deutsche übertragen: Microsoft hat ein Qualitätsproblem und dies seit Jahren. Leider gibt es sehr wenig Anzeichen dafür, dass sich dies in der nächsten Zukunft ändern wird.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/UFDyna.htm
René Probst, Juli 2000

Änderungen
- --- nil ---

Wie kann ich verhindern, dass Word2000 für jedes Dokument welches ich öffne, die Anwendung ein weiteres Mal startet?

◄ Zurück zum Hauptindex

Symptome

> 1. Wenn ich zwei Word-Dokumente z.B. vom Windows-Explorer öffne, startet Word zwei mal, das heisst auf meinem Bildschirm tummeln sich nun zwei Instanzen von Word.
>
> 2. Seit ich Word2000 einsetze, vermisse ich das Kreuz oben rechts zum Schliessen eines Dokumentes.

Ursache

Dabei handelt es sich um ein gewolltes Geschenk von Microsoft. Es soll das Umschalten zwischen den verschiedenen Word-Dokumenten für den Benutzer erleichtern und die Word-Anwendung stabiler machen. Word2000 verfügt über kein **MDI** (Multi Document Interface) mehr.

Als Konsequenz der neu gestalteten Einrichtung, erscheint oben rechts im Dokumentenfenster nicht mehr die Auswahl *Minimieren*, *Maximieren* und *Schliessen*, sondern nur noch die Schaltfläche **Schliessen.**

Lösung

Word2000 hat nur noch ein SDI (Single Document Interface). Bei anderen Office2000 Anwendungen ist dies anders, und die beschriebe Eigenart lässt sich dort abstellen. Bei Word2000 geht dies aber aus erwähnten Gründen nicht.

Anmerkungen

- Beginnen Sie morgen damit, dieses neue **Feature** von Word zu lieben. Nach einer kurzen Angewöhnungszeit von einigen Tagen, findet man es ganz Ok. Es lässt sich jetzt bequem über die Taskleiste zwischen verschiedenen Word-Dokumenten umschalten.
- Etwas lästig ist es, wenn man z. B. um eine Kleinigkeit zu testen, vier oder fünf offene Word-Fenster hat und Word nun vom Arbeitsplatz verbannen möchte. Dies um so mehr, dass natürlich für jedes Dokument weiterhin eine Rückfrage erfolgt, ob man das Dokument nun sichern oder verwerfen möchte. Sie finden etwas weiter unten ein kleines Makro, welches zur Aufgabe hat, alle Word-Instanzen auf Knopfdruck zu schliessen. Im Kopf des Codes können Sie angeben, ob eine Rückfrage erfolgen soll. Bitte beachten Sie, dass falls Sie die Rückfrage ausschalten, die entsprechenden Dokumente **nicht** gesichert werden. In diesem Falle müssten Sie schon selber veranlassen, dass die Dokumente, welche Sie gesichert haben wollen auch vorgängig gesichert wurden.
- Wenn Sie mehrere Dokumente geöffnet haben und dann den Dialog ***Datei*** aufrufen während Sie gleichzeitig die Taste **[Umschalt]** gedrückt halten, erscheint im Datei-Menü ein zusätzlicher Befehl mit dem Namen ***Alle speichern***.
- Müssig zu erwähnen, dass dieses Feature bestimmt auch den Herstellern von Speicher-Chips freuen dürfte, denn mehr Instanzen schreien nach mehr Arbeitsspeicher.
- Irgendwie kaufe ich Microsoft nicht ab, dass das MDI zu Gunsten des Benutzerkomforts fallen gelassen wurde. Vielmehr glaube ich, dass sich Microsoft während der Entwicklung von Word2000 irgend etwas eingebrockt hat, welches das tadellose funktionieren des MDI's verhindert. Ich muss allerdings gestehen, dass ich bis heute nicht dahinter gekommen bin, was den Konflikt verursachen würde.

Das Schliessen-Kreuz

Als Konsequenz des *Single Document Interfaces*, werden Sie in der Regel auch das Kreuz zum Schliessen oben rechts im Dokumentfenster nicht mehr vorfinden. Es erscheint nur noch dann, wenn genau noch ein Dokument offen ist.

Da es nun für jedes Dokument genau ein Anwendungsfenster gibt (1:1 Relation), können Sie genau so gut das Schliessen-Kreuz der Anwendung benutzen. Sollte dann noch genau ein Dokument offen sein haben Sie zwei Möglichkeiten. Sie können auch das letzte Anwendungsfenster schliessen, wobei MS Word ganz von der Oberfläche verschwindet, oder Sie können das nun vorhandene Schliessen-Kreuz anklicken, was zur Folge hat, dass das Anwendungsfenster von Word offen bleibt, aber kein Dokument mehr sichtbar ist (1:0 Relation).

Bei einer 1:1 Relation wäre das Schliessen-Kreuz im Dokumentfenster redundant zum Schliessen-Kreuz im Anwendungsfenster und würde eher irritieren, als hilfreich zu sein.

VBA-Code

```
Private Const Rückfrage = 1 '0: Mit Rückfrage; 1: Ohne Rückfrage

Sub AlleWordInstanzenSchliessen()
  Dim oApp As Task
  If Rückfrage = 0 Then
    Application.DisplayAlerts = wdAlertsNone
  Else
    Application.DisplayAlerts = wdAlertsAll
  End If
  For Each oApp In Tasks
    If InStr(oApp.Name, "Microsoft Word") Then
      oApp.Application.Quit
    End If
  Next
  Application.DisplayAlerts = wdAlertsAll
End Sub
```

(Schliesst alle aktiven Word-Instanzen)

Links zu diesem Thema

Microsoft

Englisch

► **WD2000: New Word Instance Appears to Run for Each Document**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/MDI.htm

René Probst, Oktober 1999

Änderungen

- 24. April 2000 - Umschalttaste, Alle speichern
- 29. November 2000 - Schliessen-Kreuz

Wie kann ich einzelne Abschnitte oder einzelne Seiten in Word2000 drucken?

Wie kann ich einzelne Abschnitte aus einem Seriendruckdokument drucken?

◄ Zurück zum Hauptindex

Problemstellung

In **Word 97** konnte ich einzelne Abschnitte drucken indem ich im Dialog *Drucken* z.B. den Seitenbereich *a2-a4* eintrug, was dann auch prompt ausgeführt wurde. Analog konnte ich z. B. die Vorgabe *s12a3-s15a3* eingeben und damit die Seiten 12 bis 15 im Abschnitt 3 drucken. In **Word2000** scheint dies nicht mehr zu funktionieren. Word quittiert eine solche Eingabe mit der Meldung **Ungültiger Druckbereich** und in der Folge geschieht auch kein Ausdruck. Bei anderen Gelegenheiten nimmt Word den Auftrag scheinbar entgegen, gedruckt wird aber nichts.

Lösung

Auch die deutsche Ausgabe von **Word2000** verlangt hier die englischen Begriffe, bzw. deren Abkürzungen. Der englische Begriff für Abschnitt lautet **section** und Seite heisst **page**. Nun steht die Abkürzung "s" also nicht mehr für Seite, sondern für Abschnitt! Um also die Abschnitte 4 bis 6 zu drucken, sollten Sie **s4-s6** eingeben. Um die Seiten 7 bis 10 im Abschnitt 3 zu drucken, sollten Sie folgende Eingabe machen: **p7s3-p10s3**.

Serienbrief

Angenommen, Sie haben einen einseitigen Serienbrief und möchten nun nach dem Verbinden mit der Datenquelle die Briefe für die Datensätze 20 bis 30 (inklusive) drucken, dann machen Sie im Feld **Seitenbereich - Seiten:** im Dialog ***Datei - Drucken...*** folgende Eingabe:

Word 97
s1a20-s1a30.

Word2000
p1s20-p1s30.

Anmerkungen

- Für Leute, welche mit Word programmieren, bringt es gewisse Vorteile, wenn weltweit die gleichen Begriffe bzw. die gleichen Abkürzungen zur Anwendung gelangen.
- **Achtung:** Die Word-Direkthilfe zu diesem Thema ist schlichtweg falsch.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Abschn.htm

René Probst, Oktober 1999

Änderungen
- --- nil ---

Wie kann ich eine arabische Zahl in eine römische Zahl umwandeln?

Wie kann ich eine Dezimalzahl in eine hexadezimale Zahl umwandeln?

Wie kann ich einen (Geld)-Betrag in Worte umwandeln?

◄ Zurück zum Hauptindex

Aufgabenstellungen

> 1. Ich möchte für MS Word eine Feldfunktion bzw. eine Routine, welche mir eine arabische Zahl in eine römische Zahl umwandelt.
>
> 2. Beim Ausfüllen von Checks und ähnlichen Gelegenheiten wird verlangt, dass neben der Nennung des Geldbetrages, diese Zahl auch noch in Worten ausgedrückt wird. Wie kann ich das für Word sowohl auch für Excel erreichen?
>
> 3. Ich möchte eine Dezimal Zahl in eine Hexadezimale Zahl umwandeln.

Diese Seite

- ▼ Worum geht es?
- ▼ Wie gibt man eine Feldfunktion in MS Word ein?
- ▼ Feldfunktion um eine Arabisch Zahl in eine Römische Zahl umzuwandeln
- ▼ Feldfunktion um eine Dezimalzahl in eine hexadezimale Zahl umzuwandeln
- ▼ Feldfunktion um eine Zahl (Geldbetrag) in Worten umzuwandeln
- ▼ VBA-Code zur Generierung von römischen Zahlen aus arabischen Zahlen
- ▼ VBA-Code zur Generierung von hexadezimalen Zahlen aus dezimalen Zahlen
- ▼ VBA-Code um eine Zahl (Geldbetrag) in Worte umzuwandeln
- ▼ Formatieren von arabischen Zahlen (Allgemeine numerische Formatschalter für Felder)
- ▼ Nullwerte bei Berechnungen unterdrücken
- ▼ Links zu diesem Thema

Worum geht es?

▲ Zurück zum Anfang

In diesem Skript geht es um die Umwandlung von Zahlen. Wo möglich, wird der Weg aufgezeichnet, wie Sie im Dokument eine dynamische Feldfunktion einbringen können, um Zahlen umzuwandeln. Nach dem Aufzeigen der Feldfunktion wird dann erläutert, wie das selbe mittels VBA (also mit einem Makro) zu erreichen ist.

Was die Feldfunktionen betrifft, wird die Umwandlung jeweils durch einen allgemeinen Feldschalter bewerkstelligt. Diese Schalter sind dann auch ein Schwerpunkt dieses Skriptes Sollten Sie nicht wissen, wie man einen Feldcode ganz allgemein in ein Word-Dokument einbringen kann, dann wird sie das Kapitel mit dem Titel Feldfunktionen in ein Word-Dokument einbringen interessieren.

Grundsätzlich sind diese Schalter überall dort einsetzbar, wo ein numerischer Wert zurückgeliefert wird. Es kann sich bei der Konvertierung also um eine isoliert da stehende Zahl, um das Ergebnis einer Berechnung, oder um den Inhalt eines Datenfeldes mit numerischem Feldattribut handeln.

Wie gibt man eine Feldfunktion in MS Word ein?

▲ Zurück zum Anfang

Im folgenden wird beschrieben, wie man eine Feldfunktion in ein Word-Dokument einbringt:

1. Betätigen Sie die Tastenkombination **[Alt + F9]** um die Feldcodes einzublenden.
2. Generieren Sie das Gerüst für den neuen Feldcode, indem Sie die Tastenkombination **[Strg + F9]** betätigen. Sie sollten nun ein Pärchen **geschweifter Klammern** sehen; die Einfügemarke steht mitten drin.
3. Setzen Sie nun die Einfügemarke zwischen dieses geschweifte Klammerpaar und tippen Sie die Funktion und allenfalls den Schalter wie hier beschrieben ein.
4. Betätigen Sie nun wieder die Tastenkombination **[Alt + F9]** um den Feldcode auszublenden und in die Normalansicht der Felder zurückzukehren

Feldfunktion um eine Arabisch Zahl in eine Römische Zahl umzuwandeln

▲ Zurück zum Anfang

MS Word kennt einen Feldschalter um Zahlen und Ergebnisse und anderes (zum Beispiel den Inhalt eines Seriendruckfeldes) in römischer Darstellung anzuzeigen. Es ist beides möglich, klein- und gross geschriebene römische Zahlen. Das **Maximum** liegt bei **3999**.

Wie wir alle aus der Schule wissen, kannten die alten Römer keine negativen Zahlen und auch keine 0. Falls Ihnen das Maximum von 3999 nicht genügt, weichen Sie auf die Makrolösung aus, welche in diesem Skript auch vorgestellt wird. Diese VBA Routine kann positive Zahlen bis 9'999 verarbeiten.

Römische Ziffern gross geschrieben

Die Feldfunktion um die (Jahres(zahl) 1999 in grossgeschriebenen römischen Ziffern darzustellen sieht wie folgt aus:

```
{ = 1999 \*Roman }
```

Das Resultat sieht dann wie folgt aus:

MCMXCIX

Römische Ziffern klein geschrieben

Die Feldfunktion um die (Jahres(zahl) 1999 in kleingeschriebenen römischen Ziffern darzustellen sieht wie folgt aus:

```
{ = 1999 \*roman }
```

Das Resultat sieht dann wie folgt aus:

mcmxcix

Sie können diesen Feldschalter selbstverständlich auch für das Ergebnis einer Berechnung oder z.B. für das Nachformatieren eines Seriendruckfelde, Textmarkenfeld, Word-Tabellenberechnungen, etc. gebrauchen. Hier zwei Beispiele dafür. Beachten Sie, dass **Word 97** mit deutschen Feldnamen, **Word2000** dagegen mit englischen Feldnamen arbeitet.

```
{ = 27 * 2 \*Roman }
{ MERGEFIELD Geburtsjahr \*Roman }
{ Seriendruckfeld Geburtsjahr \*Roman }
```

Feldfunktion um eine Dezimalzahl in eine hexadezimale Zahl umzuwandeln

▲ Zurück zum Anfang

MS Word kennt einen Feldschalter um Zahlen und Ergebnisse und anderes (zum Beispiel den Inhalt eines Seriendruckfeldes) in hexadezimaler Darstellung anzuzeigen. Es sind ausschliesslich gross geschriebene Hexadezimalzahlen möglich. Das **Maximum** liegt bei **32'767**.

Sollte dieses Limit nicht genügen, können Sie auf die VBA-Lösung ausweichen, welche weiter unten vorgestellt wird. Diese Funktion kann Zahlen bis zu 2'147'483'647verarbeiten.

```
{ = 2792 \*hex }
```

Das Resultat sieht dann wie folgt aus:

AE8

Sie können diesen Feldschalter selbstverständlich auch für das Ergebnis einer Berechnung oder z.B. für das Nachformatieren eines Seriendruckfelde, Textmarkenfeld, etc. gebrauchen. Hier zwei Beispiele dafür. Beachten Sie, dass Word 97 mit deutschen Feldnamen Word2000 dagegen mit englischen Feldnamen arbeitet.

```
{ = 27 * 2 \*hex }
{ MERGEFIELD IRQ \*hex }
{ Seriendruckfeld IRQ \*hex }
```

Feldfunktion um eine Zahl (Geldbetrag) in Worten umzuwandeln

▲ Zurück zum Anfang

MS Word kennt auch einen Feldschalter um Beträge, Ergebnisse und anderes (zum Beispiel den Inhalt eines Seriendruckfeldes) in Worten anzuzeigen. Lassen Sie sich nicht vom Namen des Feldfunktion irritieren. Die Funktion ist auf deutsch-deutsche Verhältnisse abgestimmt (unter der Annahme, Sie setzen die deutschsprachige Version von MS Office ein). Das **Maximum** liegt bei **999'999.99**.

Sollte sich dieses Limit als zu tief erweisen, dann setzen Sie die Makrolösung ein, welche ebenfalls in diesem Skript besprochen wird. Falls Sie Geschäfte mit der Schweiz tätigen und sich daran stören, dass die Darstellung auf deutsche Verhältnisse abgestimmt ist, dann können Sie ebenfalls auf die VBA-Lösung zurückgreifen. Diese Lösung ermöglich einerseits die Darstellung von Beträgen bis eine Billion (exklusive). Damit lässt sich sogar die private und geschäftliche Buchhaltung von Bill Gates bewältigen. Anderseits lässt sich die Funktion mit wenig Eingriffen, welche auch einem Laie gelingen, auf schweizerische Verhältnisse einstellen.

Die Unterschiede zwischen der Schweiz und Deutschland liegen übrigens darin, dass in der Schweiz dreissig (statt dreißig) geschrieben wird. Auch die Darstellung an sich ist etwas anders. Allerdings ist man in der Schweiz flexibel und nicht pingelig,; zumindest so lange, bis der Wille noch klar und unmissverständlich zur Geltung kommt.

Ein wirklich wichtiges Ausstattungsmerkmal für die Schweiz hat der VBA-Code allerdings. Er beherrscht nämlich auf Wunsch die Fünferrundung.

Negative Beträge werden nicht verarbeitet. Schreiben Sie das entsprechende Zeichen in diesem Falle vor die Feldfunktion, oder weichen Sie auf die bereits erwähnte Makro-Lösung aus.

Die Feldfunktion sieht wie folgt aus:

```
{ = 2792,95 \*Dollartext }
```

Das Resultat sieht dann wie folgt aus:

Zweitausendsiebenhundertzweiundneunzig und 95/100

Sie können diesen Feldschalter selbstverständlich auch für das Ergebnis einer Berechnung oder z.B. für das Nachformatieren eines Seriendruckfelde, Textmarkenfeld, etc. gebrauchen. Hier zwei Beispiele dafür. Beachten Sie, dass **Word 97** mit deutschen Feldnamen **Word2000** dagegen mit englischen Feldnamen arbeitet.

```
{ = 27 * 2 \*Dollartext }
{ MERGEFIELD Überweisung \*Dollartext }
{ Seriendruckfeld Überweisung \*Dollartext }
```

Beachten Sie, dass z.B. in der Schweiz ein Punkt statt ein Komma als Dezimalzeichen verwendet wird. Abhängig von den Ländereinstellung ihres Betriebssystems (Windows) müssen Sie also das richtige Dezimalzeichen verwenden, sonst quittiert Word die Feldfunktion mit dem Spruch **Syntaxfehler!**.

VBA-Code zur Generierung von römischen Zahlen aus arabischen Zahlen

▲ Zurück zum Anfang

Im folgenden wir eine allgemeingültige VBA-Routine vorgestellt, welche eine beliebige vorgegebene arabische Zahl in eine römische Zahl umwandeln kann.

Markieren Sie im Text eine ganze arabische **Zahl im Bereich von 1 bis 32'767** und führen Sie dann das folgende Makro aus. Die arabische Zahl wird dann durch das römische Equivalent ersetzt.

Die Routine ist modular aufgebaut und kann damit einfach umgebaut werden, falls Sie das Ergebnis z.B. statt im Text in einer Variablen zurück bekommen möchten.

Bevor Sie beginnen

Sie können die VBA-Routine Ihren eigenen speziellen Bedürfnissen anpassen. Dazu passen Sie im Kopf des Codes folgende Konstanten an:

- Konstante ***Gross***: Setzen Sie den Wert dieser Konstante wie folgt:
  - **0** Die römischen Ziffern erscheinen in **Kleinschreibung**.
  - **1** Die römischen Ziffern erscheinen in **Grossschreibung**.
- Konstante ***Strich***: Setzen Sie den Wert dieser Konstante wie folgt:
  - **0** Die römischen Ziffern werden **ohne** Über- und Unterstrich dargestellt.
  - **1** Die römischen Ziffern werden **mit** Über- und Unterstrich dargestellt. Dabei wird eine Feldfunktion angewendet.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
'Generelle VBA Routine, welche eine im Dokument markierte arabische
'Zahl in eine Römische umwandelt.
'
'Erstellt am 24/11/00 von René Probst / eku / CH-8152 Glattbrugg
'
'******  Anfang des anpassbaren Teils  ******
Private Const Gross = 1
Private Const Strich = 0
Private Const Titel = "Arabische Zahlen in Römische umwandeln"
'******  Ende Anpassung  ******

Sub RömischeZahl()
  If Right(Selection.Text, 1) = Chr(13) Then
```

```
    strText = Left(Selection.Text, Len(Selection.Text) - 1)
    strBurst = ""
  Else
    strText = Selection.Text
    strBurst = " "
  End If
  If Selection.Type <> 2 Or Not IsNumeric(strText) Then
    MsgBox "Markieren Sie eine Zahl.", vbInformation, Titel
    Exit Sub
  End If
  If Val(strText) < 1 Or Val(strText) > 9999 Then
    MsgBox "Die Zahl muss zwischen 1 und 9'999 liegen.", vbInformation, Titel
    Exit Sub
  End If
  If Strich = 0 Then
    Selection.Text = RömischeNummerErmitteln(Int(strText), Gross) & strBurst
    Selection.Collapse Direction:=wdCollapseEnd
  Else
    Text = " \X \to \bo (" & RömischeNummerErmitteln(strText, Gross) & ")"
    ActiveDocument.Fields.Add Range:=Selection.Range, _
       Type:=wdFieldFormula, Text:=Text, PreserveFormatting:=False
  End If
End Sub

Private Function RömischeNummerErmitteln(ByVal Nummer As Integer, Gross As Integer)
  If Nummer = 0 Then
    RN = 0
    Exit Function
  End If
  Pa = Format(Nummer, "0000")
  Dim Digit(4) As Integer
  For j = 1 To 4
    Digit(j) = Mid(Pa, j, 1)
  Next j
  Select Case Digit(1)
    Case 0
    Case Else
      For k = 1 To Digit(1)
        RN = RN + "M"
      Next k
  End Select
  Select Case Digit(2)
    Case 0
    Case 1, 2, 3
      For k = 1 To Digit(2)
        RN = RN + "C"
      Next k
    Case 4
      RN = RN + "CD"
    Case 5
      RN = RN + "D"
    Case 6, 7, 8
      RN = RN + "D"
```

```
      For k = 1 To Digit(2) - 5
        RN = RN + "C"
      Next k
    Case 9
      RN = RN + "CM"
  End Select
  Select Case Digit(3)
    Case 0
    Case 1, 2, 3
      For k = 1 To Digit(3)
        RN = RN + "X"
      Next k
    Case 4
      RN = RN + "XL"
    Case 5
      RN = RN + "L"
    Case 6, 7, 8
      RN = RN + "L"
      For k = 1 To Digit(3) - 5
        RN = RN + "X"
      Next k
    Case 9
        RN = RN + "XC"
  End Select
  Select Case Digit(4)
    Case 0
    Case 1, 2, 3
      For k = 1 To Digit(4)
        RN = RN + "I"
      Next k
    Case 4
      RN = RN + "IV"
    Case 5
      RN = RN + "V"
    Case 6, 7, 8
      RN = RN + "V"
      For k = 1 To Digit(4) - 5
        RN = RN + "I"
      Next k
    Case 9
      RN = RN + "IX"
  End Select
  If Gross = 0 Then RN = LCase(RN)
  RömischeNummerErmitteln = RN
End Function
```

(Wandelt eine markierte ganze arabische Zahl in eine römische Zahl um.)

VBA-Code zur Generierung von hexadezimalen Zahlen aus dezimalen Zahlen

▲ Zurück zum Anfang

Im folgenden wird eine allgemeingültige VBA-Routine vorgestellt, welche eine beliebige vorgegebene Dezimalzahl in eine hexadezimale Zahl umwandeln kann.

Markieren Sie im Text eine ganze dezimale **Zahl im Bereich von 0 bis 2'147'483'647** und führen Sie dann das folgende Makro aus. Die arabische Zahl wird dann durch das hexadezimale Equivalent ersetzt.

Die Routine ist modular aufgebaut und kann damit einfach umgebaut werden, falls Sie das Ergebnis z.B. statt im Text in einer Variablen zurück bekommen möchten.

Bevor Sie beginnen

Sie können die VBA-Routine Ihren eigenen speziellen Bedürfnissen anpassen. Dazu passen Sie im Kopf des Codes folgende Konstanten an:

- Konstante ***Gross***: Setzen Sie den Wert dieser Konstante wie folgt:
  - **0** Die Hexadezimalzahl erscheint in **Kleinschreibung**.
  - **1** Die Hexadezimalzahl erscheint in **Grossschreibung**.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
'******  Anfang des anpassbaren Teils  ******
Private Const Gross = 1 '0 oder 1
'******  Ende Anpassung  ******
Private Const Titel = "Dezimalzahl in Hexadezimalzahl umwandeln"

Sub Hexadezimal()
  If Right(Selection.Text, 1) = Chr(13) Then
    strText = Left(Selection.Text, Len(Selection.Text) - 1)
    strBurst = ""
  Else
    strText = Selection.Text
    strBurst = " "
  End If
  If Selection.Type <> 2 Or Not IsNumeric(strText) Then
    MsgBox "Markieren Sie eine Zahl.", vbInformation, Titel
    Exit Sub
  End If
  If strText < 0 Or Val(strText) > 2147483647 Then
    MsgBox "Die Zahl muss zwischen 0 und 2'147'483'647 liegen.", vbInformation, Titel
    Exit Sub
  End If
  Selection.Text = HexadezimalzahlErmitteln(Int(strText), Gross) & strBurst
  Selection.Collapse Direction:=wdCollapseEnd
End Sub

Private Function HexadezimalzahlErmitteln(Dezimal As Long, Gross As Integer)
  HexadezimalzahlErmitteln = Hex(Dezimal)
  If Gross = 0 Then HexadezimalzahlErmitteln = LCase(HexadezimalzahlErmitteln)
End Function
```

(Wandelt eine markierte ganze Dezimalzahl in eine Hexadezimalzahl um.)

VBA-Code um eine Zahl (Geldbetrag) in Worte umzuwandeln

▲ Zurück zum Anfang

Folgende VBA-Routine wandelt eine im Dokument markierte Zahl in Worten um. Dabei wird der markierte Betrag mit der vom Programm gebildeten Zeichenfolge ersetzt.

Bevor Sie beginnen

Sie können die VBA-Routine Ihren eigenen speziellen Bedürfnissen anpassen. Dazu passen Sie im Kopf des Codes bitte folgende Konstanten an:

Allgemeine Einstellungen

- Konstante ***Hundertstel:*** steuert das Anzeigen von Pfennigen-, Groschen- oder Rappenbeträgen. Setzen Sie den Wert dieser Konstante wie folgt fest:
  - **0** Pfennige/Groschen/Rappen werden **nicht angezeigt**.
  - **1** Pfennige/Groschen/Rappen werden angezeigt.
  - **2** Pfennige/Groschen/Rappen werden angezeigt, allerdings **nicht bei null.**
- Konstante ***Prozent:*** steuert das Format der Anzeige von Pfennigen, Groschen oder Rappen. Setzen Sie den Wert dieser Konstante wie folgt:
  - **0** Pfennige/Groschen/Rappen werden wie folgt angezeigt: **00/100** also z.B. *Dreihundertneunzehn 95/100*.
  - **1** Pfennige/Groschen/Rappen werden wie folgt angezeigt: **%00** also z.B. *Dreihundertneunzehn %95*.
- Konstante ***Vorzeichen:*** legt fest, ob bei negativen Beträgen ein Minuszeichen angezeigt wird:
  - **0** Minuszeichen wird **nicht angezeigt**.
  - **1** Minuszeichen wird angezeigt.
- Konstante: ***GSchreibung*** legt fest, ob der erste Buchstaben im Wort gross geschrieben werden soll:
  - **0** Erster Buchstabe wird **klein** geschrieben.
  - **1** Erster Buchstabe wird **gross** geschrieben.

Länderspezifische Eintellungen

- Konstante: ***DZeichen*** legt das Dezimalzeichen fest:
  - **,** Als Dezimalzeichen wird ein **Komma** erwarte, wie dies in Deutschland und Österreich üblich ist.
  - **.** Als Dezimalzeichen wird ein **Punkt** erwarte, wie dies in der Schweiz üblich ist.
- Konstante ***Dreissig*** legt fest, wie die Zahl 30 in Worten geschrieben wird:
  - **Dreißig** Die Zahl 30 wird mit dem Wort **Dreißig** ausgedrückt, wie dies in Deutschland und Österreich üblich ist.
  - **Dreissig** Die Zahl 30 wird mit dem Wort **Dreissig** ausgedrückt, wie dies in Schweiz üblich ist.
- Konstante: ***Fünfer*** legt fest, ob die Rundung auf Einer oder Fünfer geschieht. Legen Sie diesen Wert wie folgt fest:
  - **0** Die Rundung geschieht auf **ein** Pfennig bzw. ein Groschen genau.
  - **1** Die Rundung geschieht auf **fünf** (Rappen) genau, wie dies in Schweiz üblich ist.

VBA-Code

Kopieren Sie den folgenden Code in ein neues Modul der Standardvorlage **Normal.dot**. Speichern Sie die Vorlage und erstellen Sie dann ein neues Dokument. Schreiben und markieren Sie nun eine Zahl und führen sie dann das Makro aus.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
' Diese Word 97/2k-Funktion gibt (Geld)beträge in Worten aus
' Erstellt am 11/02/99 von René Probst / eku / CH-8152 Glattbrugg
'
' Der Name des Makro lautet: BetragInWorteUmwandeln
' Markieren Sie einen Betrag in Ihrem Dokument und rufen Sie die Routine auf.
'
' Verarbeitet Beträge zwischen -999'999'999 und 999'999'999
'
' * * * * * * * * * Landspezifische Einstellungen * * * * * * * * * *
Private Const DZeichen = ","       'Dezimalzeichen
Private Const Dreissig = "dreißig" 'dreissig = CH; dreißig = D / A
Private Const Fünfer = 0           '0: auf ein Pfennig/Groschen genau;
'                                   1: auf fünf Rappen genau
' * * * * * * * * * Ende Landspezifische Einstellungen * * * * * * * *
'
Private Const Hundertstel = 1 '0: Pfennige/Groschen/Rappen werden nicht einbezogen;
'                              1: werden einbezogen
'                              2: werden einbezogen aber nicht bei 00
Private Const Prozent = 0     '0: Format 00/100;
'                              1 = % plus Nachkommastellen
Private Const Vorzeichen = 1  '0: Minus als Vorzeichen wird nicht gezeigt;
'                              1: wird angezeigt
Private Const GSchreibung = 1 '0: 1. Buchstaben klein geschrieben;
'                              1: 1. Buchstaben gross geschrieben
Dim Wort As String

Private Const Titel = "Zahlen in Worten"

Sub BetragInWorteUmwandeln()
  If Selection.Type <> wdSelectionNormal Then
    MsgBox "Markieren Sie einen Betrag.", , Titel
    End
  End If
  x = Selection.Text
  If Right(x, 1) = Chr(13) Then x = Left(x, Len(x) - 1)
  If Not IsNumeric(x) Then
    MsgBox "Der markierte Text ist kein gültiger Betrag.", , Titel
    End
  End If
  Selection.Text = InWorten(x)
End Sub

Private Function InWorten(ByVal Betrag As Double) As String
  Vorz = ""
  If Vorzeichen = 1 And Betrag < 0 Then Vorz = "- "
```

```
  If Abs(Betrag) > (1000000000 - 0.006) Then
    InWorten = "Ungültig!"
    Exit Function
  End If
  If Betrag < 0 Then Betrag = Betrag * -1
  ZahlenInWorten Betrag
  InWorten = Vorz & Wort
  If GSchreibung = 1 Then
    InWorten = Vorz & UCase(Left(Wort, 1)) + Right(Wort, Len(Wort) - 1)
  End If
End Function

Private Sub ZahlenInWorten(Betrag)
  Wort = ""
  GanzBetrag = Betrag
  Betrag = Format(Betrag, "#.00")
  If InStr(Betrag, DZeichen) > 0 Then Betrag = Left(Betrag, InStr(Betrag, DZeichen) -
1)
  Länge = Len(Betrag)
  Max = Länge \ 3
  tmp = Länge Mod 3
  If tmp > 0 Then Max = Max + 1
  ReDim Tbetr(Max)
  tmp = Betrag
  For i = Max To 1 Step -1 'Bilden von 3-er Paketen von rechts aus gesehen
    Tbetr(i) = Right(tmp, 3)
    If Len(tmp) > 3 Then tmp = Left(tmp, Len(tmp) - 3)
  Next i
  i = 1
  For i = 1 To Max ' Alle 3-er Pakete verarbeiten
    Lg = Len(Tbetr(i))
    If i = Max Then Flag = True Else Flag = False
    If Lg = 1 Then Wort = GetEiner(Right(Tbetr(i), 1))
    If Lg = 2 Then Wort = Wort + GetZehner(Mid(Tbetr(i), Lg - 1, 1), Tbetr(i))
    If Lg = 3 Then Wort = Wort + GetHunderter(Mid(Tbetr(i), Lg - 2, 1), Tbetr(i), Flag)
    If Tbetr(i) > 0 And Betrag > 999 Then
      If (Max - i) = 1 Then Wort = Wort + "tausend"
      If (Max - i) = 2 Then Wort = Wort + "millionen"
    End If
  Next i
  If InStr(Wort, "einmillionen") = 1 Then
    Wort = "einemillion" + Right(Wort, Len(Wort) - 12)
  End If
  If Right(Wort, 3) = "ein" Then Wort = Left(Wort, Len(Wort) - 3) + "eins"
  If Hundertstel > 0 Then Wort = Wort + GetHundertstel(GanzBetrag)
End Sub

Private Function GetEiner(Zahl)
  Select Case Zahl
    Case 0: Z = "null"
    Case 1: Z = "ein"
    Case 2: Z = "zwei"
    Case 3: Z = "drei"
    Case 4: Z = "vier"
```

```
    Case 5: Z = "fünf"
    Case 6: Z = "sechs"
    Case 7: Z = "sieben"
    Case 8: Z = "acht"
    Case 9: Z = "neun"
  End Select
  GetEiner = Z
End Function

Private Function GetZehner(Zahl, Betrag)
  Betrag = Right(Betrag, 2)
  Select Case Betrag
    Case 10: Z = "zehn"
    Case 11: Z = "elf"
    Case 12: Z = "zwölf"
    Case 13, 14, 15, 18, 19: Z = GetEiner(Right(Betrag, 1)) + "zehn"
    Case 16: Z = "sechzehn"
    Case 17: Z = "siebzehn"
    Case 20: Z = "zwanzig"
    Case 21 To 29: Z = GetEiner(Right(Betrag, 1)) + "undzwanzig"
    Case 30: Z = Dreissig
    Case 31 To 39: Z = GetEiner(Right(Betrag, 1)) + "und" + Dreissig
    Case 40: Z = "vierzig"
    Case 41 To 49: Z = GetEiner(Right(Betrag, 1)) + "undvierzig"
    Case 50: Z = "fünfzig"
    Case 51 To 59: Z = GetEiner(Right(Betrag, 1)) + "undfünfzig"
    Case 60: Z = "sechzig"
    Case 61 To 69: Z = GetEiner(Right(Betrag, 1)) + "undsechzig"
    Case 70: Z = "siebzig"
    Case 71 To 79: Z = GetEiner(Right(Betrag, 1)) + "undsiebzig"
    Case 80: Z = "achtzig"
    Case 81 To 89: Z = GetEiner(Right(Betrag, 1)) + "undachtzig"
    Case 90: Z = "neunzig"
    Case 91 To 99: Z = GetEiner(Right(Betrag, 1)) + "undneunzig"
  End Select
  GetZehner = Z
End Function

Private Function GetHunderter(Zahl, Betrag, Flag)
  If Zahl > 0 Then Y = GetEiner(Zahl) + "hundert"
  Select Case Right(Betrag, 2)
    Case 0: Z = ""
    Case 1 To 9: Z = GetEiner(Right(Betrag, 1))
    Case Else: Z = GetZehner(Zahl, Betrag)
  End Select
  If Flag = True And Betrag > 0 And Zahl = 0 Then
    GetHunderter = "und" + Y + Z
  Else
    GetHunderter = Y + Z
  End If
End Function

Private Function GetHundertstel(Betrag)
```

```
  If Fünfer = 1 Then
    Betrag = Format((Betrag * 20 + 0.5) / 20, "#.00")
  Else
    Betrag = Format(Betrag, "#.00")
  End If
  If Prozent = 1 Then
    P1 = "%"
    P2 = ""
  Else
    P1 = " "
    P2 = "/100"
  End If
  GetHundertstel = P1 + Right(Betrag, 2) + P2
End Function
```

(Eine markierte Zahl wird *in Worte* umgewandelt)

Zahlen in einem Serienbrief in Worte umwandeln

Leicht modifiziert, lässt sich diese Routine auch auf Serienbriefe anwenden. Allerdings ist es leider so, dass die Word-Serienbrieffunktion keine Möglichkeit bietet, während des Zusammenführens der Daten mit dem Hauptdokument für jeden Datensatz das Makro, also eine Art Serienbrief-Feld-Exit-Funktion, aufzurufen. Hier zeigt sich, dass MS Word in dieser Beziehung etwas schwach auf der Brust ist, denn bei manch anderen Systemen ist eine solche Funktion selbstverständlich und naheliegend.

Bleibt uns die Option, die hier beschriebene Funktion über das Resultat, also über die fertigen Serienbriefe laufen zu lassen. Das ist eine durchaus praktikable Lösung und sollte eigentlich gelingen.

Dazu leiten Sie den Serienbrief nicht direkt zum Drucker, sondern in ein neues Dokument. Anschliessend lassen sie die folgende, leicht abgeänderte Variante des Makro, über dieses Dokument laufen. Anschliessend können Sie das Dokument, oder Teile davon, ganz konventionell ausdrucken.

Gehen Sie im Einzelnen wie folgt vor:

1. Kopieren Sie den unten abgedruckte Code in das Serienbriefhauptdokument.
2. Passen Sie im Kopf des Codes die Konstanten nach Ihren speziellen Bedürfnissen an, und zwar in der selben Weise, wie dies im vorhergehenden Kapitel ausführlich beschrieben wurde.
3. Fügen Sie in das Hauptdokument, das Feld, welches den Betrag beinhaltet und das in Worte umgewandelt werden soll, nochmals ein. Diesmal aber eingebettet in Dollar-Zeichen. Also **drei** Dollar- Zeichen, dann das Feld und dann nochmals **drei** Dollar-Zeichen. Diese Definition soll keinerlei Leerschläge aufweisen und sieht dann etwa wie folgt aus:

   Bitte Überweisen Sie den Betrag von "Betrag" (in Worten $$$"Betrag"$$$) in den nächsten Tagen auf mein Konto.

   Dieses Beispiel geht von der Annahme aus, dass das Feld, welches umgewandelt werden soll den Namen *Betrag* hat. Wählen Sie hier den effektiven Name des Feldes, also so wie dieser in Ihrer Datenbank oder Ihrer Excel-Tabelle wirklich heisst.
4. Führen Sie nun den Seriendruck aus. Wählen Sie dabei die Option **Ausgabe in ein neues Dokument**.
5. Bringen Sie nun das Dokument mit dem Ergebnis in den Vordergrund (also **nicht** das Serienbriefhauptdokument) und führen Sie das Makro mit dem Name **PostSeriendruck** aus.

```
Private Const DZeichen = ","       'Dezimalzeichen
Private Const Dreissig = "dreißig" 'dreissig = CH; dreißig = D / A
Private Const Fünfer = 0           '0: auf ein Pfennig/Groschen genau;
'                                   1: auf fünf Rappen genau
' * * * * * * * * * Ende Landspezifische Einstellungen * * * * * * * *
'
Private Const Hundertstel = 1 '0: Pfennige/Groschen/Rappen werden nicht einbezogen;
'                              1: werden einbezogen
'                              2: werden einbezogen aber nicht bei 00
Private Const Prozent = 0     '0: Format 00/100;
'                              1 = % plus Nachkommastellen
Private Const Vorzeichen = 1  '0: Minus als Vorzeichen wird nicht gezeigt;
'                              1: wird angezeigt
Private Const GSchreibung = 1 '0: 1. Buchstaben klein geschrieben;
'                              1: 1. Buchstaben gross geschrieben
Dim Wort As String

Sub PostSeriendruck()
  Application.ScreenUpdating = False
  Selection.HomeKey Unit:=wdStory
  With Selection.Find
    .ClearFormatting
    .Text = "$$$*$$$"
    .MatchWildcards = True
    .Execute
    While .Found = True
      If Selection.Range = "$$$$$$" Then
        x = 0
      Else
        x = InWorten(Mid(Selection.Range, 4, Len(Selection.Range) - 6))
      End If
      Selection.Range = x
      .Execute
    Wend
  End With
  Application.ScreenUpdating = True
End Sub

Private Function InWorten(Betrag As Double) As String
  Vorz = ""
  If Vorzeichen = 1 And Betrag < 0 Then Vorz = "- "
  If Abs(Betrag) > (1000000000 - 0.006) Then
    InWorten = "Ungültig!"
    Exit Function
  End If
  If Betrag < 0 Then Betrag = Betrag * -1
  ZahlenInWorten Betrag
  InWorten = Vorz & Wort
  If GSchreibung = 1 Then
    InWorten = Vorz & UCase(Left(Wort, 1)) + Right(Wort, Len(Wort) - 1)
  End If
End Function
```

```
Private Sub ZahlenInWorten(Betrag)
  Wort = ""
  GanzBetrag = Betrag
  Betrag = Format(Betrag, "#.00")
  If InStr(Betrag, DZeichen) > 0 Then Betrag = Left(Betrag, InStr(Betrag, DZeichen) - 1)
  Länge = Len(Betrag)
  Max = Länge \ 3
  tmp = Länge Mod 3
  If tmp > 0 Then Max = Max + 1
  ReDim Tbetr(Max)
  tmp = Betrag
  For i = Max To 1 Step -1 'Bilden von 3-er Paketen von rechts aus gesehen
    Tbetr(i) = Right(tmp, 3)
    If Len(tmp) > 3 Then tmp = Left(tmp, Len(tmp) - 3)
  Next i
  i = 1
  For i = 1 To Max ' Alle 3-er Pakete verarbeiten
    Lg = Len(Tbetr(i))
    If i = Max Then Flag = True Else Flag = False
    If Lg = 1 Then Wort = GetEiner(Right(Tbetr(i), 1))
    If Lg = 2 Then Wort = Wort + GetZehner(Mid(Tbetr(i), Lg - 1, 1), Tbetr(i))
    If Lg = 3 Then Wort = Wort + GetHunderter(Mid(Tbetr(i), Lg - 2, 1), Tbetr(i), Flag)
    If Tbetr(i) > 0 And Betrag > 999 Then
      If (Max - i) = 1 Then Wort = Wort + "tausend"
      If (Max - i) = 2 Then Wort = Wort + "millionen"
    End If
  Next i
  If InStr(Wort, "einmillionen") = 1 Then
    Wort = "einemillion" + Right(Wort, Len(Wort) - 12)
  End If
  If Right(Wort, 3) = "ein" Then Wort = Left(Wort, Len(Wort) - 3) + "eins"
  If Hundertstel > 0 Then Wort = Wort + GetHundertstel(GanzBetrag)
End Sub

Private Function GetEiner(Zahl)
  Select Case Zahl
    Case 0: Z = "null"
    Case 1: Z = "ein"
    Case 2: Z = "zwei"
    Case 3: Z = "drei"
    Case 4: Z = "vier"
    Case 5: Z = "fünf"
    Case 6: Z = "sechs"
    Case 7: Z = "sieben"
    Case 8: Z = "acht"
    Case 9: Z = "neun"
  End Select
  GetEiner = Z
End Function

Private Function GetZehner(Zahl, Betrag)
  Betrag = Right(Betrag, 2)
  Select Case Betrag
    Case 10: Z = "zehn"
```

```
    Case 11: Z = "elf"
    Case 12: Z = "zwölf"
    Case 13, 14, 15, 18, 19: Z = GetEiner(Right(Betrag, 1)) + "zehn"
    Case 16: Z = "sechzehn"
    Case 17: Z = "siebzehn"
    Case 20: Z = "zwanzig"
    Case 21 To 29: Z = GetEiner(Right(Betrag, 1)) + "undzwanzig"
    Case 30: Z = Dreissig
    Case 31 To 39: Z = GetEiner(Right(Betrag, 1)) + "und" + Dreissig
    Case 40: Z = "vierzig"
    Case 41 To 49: Z = GetEiner(Right(Betrag, 1)) + "undvierzig"
    Case 50: Z = "fünfzig"
    Case 51 To 59: Z = GetEiner(Right(Betrag, 1)) + "undfünfzig"
    Case 60: Z = "sechzig"
    Case 61 To 69: Z = GetEiner(Right(Betrag, 1)) + "undsechzig"
    Case 70: Z = "siebzig"
    Case 71 To 79: Z = GetEiner(Right(Betrag, 1)) + "undsiebzig"
    Case 80: Z = "achtzig"
    Case 81 To 89: Z = GetEiner(Right(Betrag, 1)) + "undachtzig"
    Case 90: Z = "neunzig"
    Case 91 To 99: Z = GetEiner(Right(Betrag, 1)) + "undneunzig"
  End Select
  GetZehner = Z
End Function

Private Function GetHunderter(Zahl, Betrag, Flag)
  If Zahl > 0 Then Y = GetEiner(Zahl) + "hundert"
  Select Case Right(Betrag, 2)
    Case 0: Z = ""
    Case 1 To 9: Z = GetEiner(Right(Betrag, 1))
    Case Else: Z = GetZehner(Zahl, Betrag)
  End Select
  If Flag = True And Betrag > 0 And Zahl = 0 Then GetHunderter = "und" + Y + Z Else
GetHunderter = Y + Z
End Function

Private Function GetHundertstel(Betrag)
  If Fünfer = 1 Then
    Betrag = Format((Betrag * 20 + 0.5) / 20, "#.00")
  Else
    Betrag = Format(Betrag, "#.00")
  End If
  If Prozent = 1 Then
    P1 = "%"
    P2 = ""
  Else
    P1 = " "
    P2 = "/100"
  End If
  GetHundertstel = P1 + Right(Betrag, 2) + P2
End Function
```

(Das Ergebnis eines Seriendruckes wird nachbearbeitet mit dem Ziel einen Betrag *in Worte* umzuwandeln.)

Formatieren von arabischen Zahlen (Allgemeine numerische Formatschalter für Felder)

▲ Zurück zum Anfang

Mit seinen numerischen Feldschaltern bietet MS Word die Möglichkeit, das Aussehen einer Zahl, welche als Feldfunktion eingebracht wurde zu beeinflussen. Dabei handelt es sich nicht um eine nominelle Veränderung der Zahl; vielmehr wird das Erscheinungsbild gesteuert. Die folgende Tabelle zeigt Formatierungsoptionen, welche in der Praxis häufig gefragt sind, auf.

Einmal mehr gilt: Sie können diese Schalter auf nackte Zahlen, Ergebnisse aus Berechnungen oder Datenfelder (z.B. Seriendruckfelder) einwirken lassen.

Formatierung	Syntax	Beispiele	Anzeige	Bemerkungen

Führende Nullen	`\# "0"`
	`\# "00"`
	`\# "000"`
	`etc.`
		`{ = 4058 \# "00000"}`
		`{ = 4 + 5 \# "0000"}`
		`{ Mergefield \# "0000"}`
			04058
			0009
			xxxx
				--- 	

Dezimalstellen	`\# "0.00"`
	`\# "0.000"`
	`\# "0.000"`
	`etc.`
		`{ = 3.2 \# "00.00"}`
		`{ = 4 + 5 \# "00.00"}`
		`{ = 3 / 4 \# ".00"}`
		`{ = 3 / 4 \# ".0"}`
		`{ Mergefield \# "00.00"}`
			03.20
			09.00
			.75
			.8
			xx.xx
				Das Dezimaltrennzeichen kann variieren und hängt von den Ländereinstellungen unter Windows ab.
				In Deutschland kommt das Komma zum Zuge, in der Schweiz ein Punkt.
				Bei Bedarf wird das Ergebnis kaufmännisch gerundet.
				Die Fünferrundung, welche in der Schweiz gefragt ist, beherrscht leider keiner dieser Feldschalter.

Ausrichtung	`\# "#.00"`
	`\# "##.00"`
	`\# "###.00"`
	`\# "#'###.00"`
	`etc.`
		`{ = 125/25 \# "#'###.00" }`	5.00	Die Ausrichtung erfolgt am Dezimalzeichen. Links wird mit Leerstellen ausgefüllt
				Bei Bedarf wird das Tausendertrennzeichen beigefügt.
				Die Ausrichtung gelingt nur mit einer nicht proportionalen Schrift wirklich gut.

Tausendertrennzeichen	`\# "#'###'###"`	`{ = 1234567890 \# "#'###'###" }`	1'234'567'890	Lässt sich mit dem Dezimalzeichen auch kombinieren. Das Tausendertrennzeichen kann variieren und hängt von den Ländereinstellungen unter Windows ab.
Beträge, Währungen	`\# "DM #'###.##"` `\# "DM #'###.00"`	`{ = 666.806 \# "DM #'###.##" }` `{ = 1234 \# "DM #'###.00" }`	DM 666.81	Es können selbstverständlich auch andere Währungssymbole eingesetzt werden. Es findet eine kaufmännische Rundung statt. Das Tausendertrennzeichen kann variieren und hängt von den Ländereinstellungen unter Windows ab.
Nullwert unterdrücken	`\# "0,00;-0,00;''"}`	`{ = Produkt(a2;c2) \# "0,00;-0,00;''"}`	---	Wenn das Resultat positiv ist erscheint ein positives Ergebnis, bei einem negativen Resultat ein negatives Ergebnis und bei 0 erscheint nichts.
Prozent	`\# "##%"`	`{ = 33 \# "##%" }`	33%	---

Nullwerte bei Berechnungen unterdrücken

▲ Zurück zum Anfang

Lesen Sie dazu das entsprechende Skript, dessen Link gleich anschliessend eingeblendet ist.

Wie gibt man eine Feldfunktion in MS Word ein?

▲ Zurück zum Anfang

Im folgenden wird beschrieben, wie man eine Feldfunktion in ein Word-Dokument einbringt:

1. Betätigen Sie die Tastenkombination **[Alt + F9]** um die Feldcodes einzublenden.
2. Generieren Sie das Gerüst für den neuen Feldcode, indem Sie die Tastenkombination **[Strg + F9]** betätigen. Sie sollten nun ein Pärchen **geschweifter Klammern** sehen; die Einfügemarke steht mitten drin.
3. Setzen Sie nun die Einfügemarke zwischen dieses geschweifte Klammerpaar und tippen Sie die Funktion und allenfalls den Schalter wie hier beschrieben ein.
4. Betätigen Sie nun wieder die Tastenkombination **[Alt + F9]** um den Feldcode auszublenden und in die Normalansicht der Felder zurückzukehren

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Die Funktion InWorten als Excel-Lösung
► Wie kann ich Nullwerte beim Berechnung unterdrücken?

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Zahlen.htm

René Probst, März 2000

Änderungen

- 24. November 2000 - Übernommen von Roemisch.htm und InWorten.htm
- 24. November 2000 - Feldfunktionen hinzugefügt
- 24. November 2000 - Feldfunktion Dollartext
- 24. November 2000 - Hexadezimalzahl

Wie kann ich ein Beispieltext für alle vorhandenen Schriften anzeigen oder ausdrucken?

◄ Zurück zum Hauptindex

Aufgabenstellung

> Ich möchte mir einen Überblick verschaffen wie die Schriften, welche in MS Word wählbar sind, aussehen bzw. wirken.

Lösung

Der folgende VBA-Code erstellt in Word 97 oder Word2000 eine Tabelle mit allen verfügbaren Schriften. Dazu wird für jede Schrift ein Beispieltext gedruckt.

VBA-Code

```
Private Const Probetext = _
   "Franz jagt im komplett verwahrlosten Taxi quer durch Bayern. 0123456789"

Sub SchriftenGeneratorI()
  Set rDoc = Documents.Add
  System.Cursor = wdCursorWait
  Application.ScreenUpdating = False
  Selection.TypeText "Name" & vbTab & "Schriftprobe" & vbCrLf
  For Each Schrift In Application.FontNames
    With Selection
      .Font.Name = "Arial"
      .Font.Size = 10
      .TypeText Text:=Schrift & vbTab
      .Font.Name = Schrift
      .TypeText Text:=Probetext & vbCrLf
    End With
  Next
  With Selection
    .WholeStory
    .ConvertToTable Separator:=wdSeparateByTabs, _
       Format:=wdTableFormatProfessional, AutoFit:=True
    .Sort ExcludeHeader:=True
    .ParagraphFormat.SpaceBefore = 2
    .ParagraphFormat.SpaceAfter = 2
    .HomeKey Unit:=wdStory
    .SplitTable
    .Style = wdStyleHeading1
    .TypeText Text:="Schriftproben"
    .EndKey Unit:=wdStory
    .Style = wdStyleNormal
    .HomeKey Unit:=wdStory
  End With
  Application.ScreenUpdating = True
  System.Cursor = wdCursorNormal
  Set oAssistant = Assistant
  If Not oAssistant Is Nothing Then
    Assistant.Animation = msoAnimationCharacterSuccessMajor
  End If
  Set oAssistant = Nothing
End Sub
```

(Der Schriftprobengenerator I - Druckt als Beispieltext den Satz: "Franz jagt im komplett verwahrlosten Taxi quer durch Bayern. 0123456789")

Hier eine Alternative dazu. Es werden alle potentiell Verfügbaren Zeichen im Probetext gedruckt, also auch Sonderzeichen.

```
Sub SchriftenGeneratorII()
  For i = 21 To 255
    Probetext = Probetext & Chr(i)
  Next i
  Set rDoc = Documents.Add
  System.Cursor = wdCursorWait
  Application.ScreenUpdating = False
  Selection.TypeText "Name" & vbTab & "Schriftprobe" & vbCrLf
  For Each Schrift In Application.FontNames
    With Selection
      .Font.Name = "Arial"
      .Font.Size = 10
      .TypeText Text:=Schrift & vbTab
      .Font.Name = Schrift
      .TypeText Text:=Probetext & vbCrLf
    End With
  Next
  With Selection
    .WholeStory
    .ConvertToTable Separator:=wdSeparateByTabs, _
       Format:=wdTableFormatProfessional, AutoFit:=True
    .Sort ExcludeHeader:=True
    .ParagraphFormat.SpaceBefore = 2
    .ParagraphFormat.SpaceAfter = 2
    .HomeKey Unit:=wdStory
    .SplitTable
    .Style = wdStyleHeading1
    .TypeText Text:="Schriftproben"
    .EndKey Unit:=wdStory
    .Style = wdStyleNormal
    .HomeKey Unit:=wdStory
  End With
  Application.ScreenUpdating = True
  System.Cursor = wdCursorNormal
  Set oAssistant = Assistant
  If Not oAssistant Is Nothing Then
    Assistant.Animation = msoAnimationCharacterSuccessMajor
  End If
  Set oAssistant = Nothing
End Sub
```

(Der Schriftprobengenerator II - Druckt als Beispieltext alle potentiell verfügbaren Zeichen.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/SProbe.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich eruieren, welche Schriften in einem Dokument zur Anwendung kommen?

◄ Zurück zum Hauptindex

Aufgabenstellung

> **Ich habe ein grösseres und komplex formatiertes Dokument vor mir und möchte nun wissen, welche Schriftarten das Dokument enthält.**

Lösung

Der folgende VBA-Code erstellt in Word 97 oder Word2000 eine Tabelle mit allen Schriften, welche im aktiven Dokument Verwendung finden. Der Code berücksichtigt auch Schriften, welche in Textfeldern, Fuss- und Kopfzeilen, Fussnotentext und Endnotentext zur Anwendung kommen. Das Programm listet dann auch auf, in welchem Dokumententeil eine bestimmte Schrift **zum ersten Mal** erscheint. Als Zugabe wird auch für jede gefundene Schrift eine Schriftprobe gedruckt.

Anmerkung

- Sollten Sie viele Schriften auf Ihrem Rechner installiert haben, ist die Verarbeitungsgeschwindigkeit zumindest bei grossen Dokumenten, nicht gerade berauschend.

VBA-Code

```
Private Const SampText = _
   "Franz jagt im komplett verwahrlosten Taxi quer durch Bayern. 0123456789"
Private AppFonts() As String
Private AppFontsNum As Integer
Private Part As Object
Private Hit As Boolean

Sub SchriftenAuflistung()
  ReDim Story(11) As String
  Story(1) = "Haupttext"
  Story(2) = "Fussnotentext"
  Story(3) = "Endnotentext"
  Story(4) = "Kommentare"
  Story(5) = "Textfeld"
  Story(6) = "Kopfzeilen gerade Seiten"
  Story(7) = "Kopfzeilen (ungerade Seiten)"
  Story(8) = "Fusszeilen gerade Seiten"
  Story(9) = "Fusszeilen (ungerade Seiten)"
  Story(10) = "Kopfzeile erste Seite"
  Story(11) = "Fusszeile erste Seite"
  AppFontsNum = 0
  ReDim AppFonts(Application.FontNames.Count, 1)
  Set oDoc = ActiveDocument
  System.Cursor = wdCursorWait
  Application.ScreenUpdating = False
  For Each AppFont In FontNames
    Selection.HomeKey Unit:=wdStory
    Hit = False
```

```
    SearchLoop AppFont
    If Hit = True Then
      AppFontsNum = AppFontsNum + 1
      AppFonts(AppFontsNum, 0) = AppFont
      AppFonts(AppFontsNum, 1) = Part.StoryType
    End If
  Next
  Set rDoc = Documents.Add
  Application.ScreenUpdating = False
  Selection.TypeText "Name" & vbTab & "SR" & vbTab & "Sample" & vbCrLf
  For i = 1 To AppFontsNum
    With Selection
      .Font.Name = "Arial"
      .Font.Size = 10
      .TypeText Text:=AppFonts(i, 0) & vbTab & AppFonts(i, 1) & vbTab
      .Font.Name = AppFonts(i, 0)
      .TypeText Text:=SampText & vbCrLf
    End With
  Next
  With Selection
    .WholeStory
    .ConvertToTable Separator:=wdSeparateByTabs, _
        Format:=wdTableFormatProfessional, AutoFit:=True
    If AppFontsNum > 1 Then .Sort ExcludeHeader:=True
    .ParagraphFormat.SpaceBefore = 2
    .ParagraphFormat.SpaceAfter = 2
    .HomeKey Unit:=wdStory
    .SplitTable
    .Style = wdStyleHeading1
    .TypeText Text:="Verfügbare Schriften in " & oDoc.Name
    .EndKey Unit:=wdStory
    .Style = wdStyleNormal
    .EndKey Unit:=wdStory
    .TypeParagraph
    .TypeParagraph
    For i = 1 To 11
      .TypeText Text:=i & vbTab & Story(i) & vbCrLf
    Next i
  End With
  rDoc.Content.LanguageID = wdNoProofing
  Selection.HomeKey Unit:=wdStory
  Application.ScreenUpdating = True
  System.Cursor = wdCursorNormal
  Set oAssistant = Assistant
  If Not oAssistant Is Nothing Then Assistant.Animation = _
     msoAnimationCharacterSuccessMajor
End Sub

Private Sub SearchLoop(tFont)
  For Each Part In ActiveDocument.StoryRanges
    SearchFont tFont
    While (Hit = False) And (Not (Part.NextStoryRange Is Nothing))
      Set Part = Part.NextStoryRange
```

```
      SearchFont tFont
    Wend
    If Hit = True Then Exit For
  Next
End Sub

Private Sub SearchFont(tFont)
  Selection.HomeKey Unit:=wdStory
  With Part.Find
    .ClearFormatting
    .Font.Name = tFont
    .Wrap = wdFindContinue
    .Text = ""
    .Execute
    Hit = .Found
  End With
End Sub
```

(Eruiert alle verwendeten Schriften in einem Dokument und listet diese auf. Dieser Code berücksichtigt auch die Schriften, welche ausserhalb des Haupttextes Verwendung finden.)

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/SEru.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich ein Kapitel aus einem gegliederten Dokument drucken?

◄ Zurück zum Hauptindex

Problem

> **Oft muss ich, ein einzelnes Kapitel eines umfangreichen und gegliederten Dokumentes drucken. Diese Aufgabe erweist sich als etwas mühselig, muss ich doch den Anfang des Kapitels suchen, mir die Seitenzahl merken und nochmals das selbe mit dem Ende des Kapitels tun. Dann muss ich diese Angaben im Druckdialog eintippen. Zudem vergesse ich oft die Seitenzahlen zu aktualisieren, was zur Folge hat, dass die falschen Seiten gedruckt werden und damit Zeit und Papier verschwendet wird.**

Antwort

MS Word bietet abgesehen davon, dass das ganze Dokumentes gedruckt werden kann, auch die Optionen einen Seitenbereich, einzelne Abschnitte oder im Dokument markierten Text zu drucken. Die Funktonalität, einen Druckauftrag ausgehend von einem Kapitel aufzugeben, finden Sie aber auch nach längerem Suchen nicht. Zwar scheint es aus der Gliederungsansicht bequem zu gehen, enttäuscht stellen Sie dann aber fest, dass zwar das ganze Kapitel samt seinen Unterebenen gedruckt wird, nicht aber Kopf-/ und Fusszeilen.

Im folgenden wird eine Lösung aufgezeigt, um diese Lücke zu schliessen. Dabei wird eine *UserForm* eingesetzt, welche die Hierarchie der Überschriften korrekt abbilden kann. Sie können dann ein Kapitel markieren und dieses auf Knopfdruck ausdrucken.

Kurz um, dieser Dialog ist sehr einfach zu bedienen. Schaltflächen für die *UserForm* müssen Sie keine malen. Dies wird vom Code dynamisch erledigt.

Lösung

Implementierung der Lösung

Um die folgende Lösung auf Ihrem Rechner verfügbar zu machen, gehen Sie bitte wie folgt vor:

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** (über Datei - Öffnen)
2. Wechseln Sie mit der Tastenkombination **[Strg + F11]** in die VBA-Umgebung.
3. Wählen Sie im VBA-Editor im Hauptmenü den Dialog ***Einfügen - Modul*** und kopieren Sie den kurzen Code (3-Zeiler), welcher unten abgebildet ist, in das leere Code-Fenster.
4. Wählen Sie nun im Hauptmenü den Dialog ***Einfügen - UserForm***.
5. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt **Code anzeigen**.
6. Löschen Sie nun die zwei Code-Zeilen, welche von Word automatisch generiert wurden und kopieren Sie den grösseren Code, welcher unten abgebildet ist in das nun leere Fenster.
7. Lassen Sie sich nun mit der Taste **F4** den Eigenschaftsdialog der UserForm anzeigen.
8. Ändern Sie nun die Eigenschaft **Name** von *UserFormx* nach **frmKapitelDrucken**.
9. Kehren Sie in das Dokumentfenster zurück.
10. Wählen Sie dort vom Hauptmenü den Dialog ***Extras - Anpassen...***.
11. Aktivieren Sie den Reiter **Befehle**.
12. Markieren Sie in der linken Liste mit der Bezeichnung *Kategorien:* den Eintrag **Makros**.
13. Suchen Sie in der rechten Liste mit der Bezeichnung *Befehle* den Eintrag **KapitelDrucken**.
14. Ziehen Sie diesen Eintrag mit der Maus in das Menü **Datei**, welches aufklappen sollte, wenn sich der Mauszeiger nähert und legen Sie die Schaltfläche dort ab.
15. Optional können Sie immer noch in diesem Anpassungsmodus einen Rechtsklick auf die Schaltfläche ausüben, und über das Kontextmenü Name und Aussehen der Schaltfläche beeinflussen.
16. Speichern Sie nun die *Normal.dot* und verlassen Sie Word.
17. Nachdem Sie Word wieder gestartet habe, können Sie die neue Funktion nutzen.

VBA-Code

```
Sub KapitelDrucken()
  frmKapitelDrucken.Show
End Sub
```

(Ruft die UserForm, welche dem Drucken von Kapiteln dient, auf)

```
'Dieser Word 97/2000-Dialog ermöglicht es ein Kapitel eines
'gegliederten Dokumentes, auf bequeme Art und Weise zu drucken.
'
'Erstellt am 11/11/00 von René Probst / eku / CH-8152 Glattbrugg
'
'Die Vervielfältigung (auch auf Datenträgern aller Art), die
'Veröffentlichung und der Vertrieb mit kommerziellem Motiv ist
'untersagt!
'
'******* Anfang des anpassbaren Teils *********
Private Const Titel = "Kapitel drucken"
'******* Ende des anpassbaren Teils *********

Private WithEvents lbOutline As MSForms.ListBox
Private WithEvents spnLevels As MSForms.SpinButton
Private WithEvents cmdGeheZu As MSForms.CommandButton
Private WithEvents cmdDrucken1 As MSForms.CommandButton
Private WithEvents cmdDrucken2 As MSForms.CommandButton
Private WithEvents cmdSchliessen As MSForms.CommandButton
Private lblBereich As MSForms.Label
Private lblDrucker As MSForms.Label
Private txtLevels As MSForms.TextBox
Private oDoc As Document
Private hd() As String
Private ol() As String
Private Ebenen As Long
Public savEnv As Boolean

Private Sub UserForm_Initialize()
  Set oDoc = ActiveDocument
  Me.Width = 310
  Me.Height = 230
  Me.Caption = Titel
  Me.Tag = "*INIT"
  If HeadingsSammeln > 0 Then End
  SchaltflaechenErstellen
  spnLevels.Min = 1
  spnLevels.Max = Ebenen
  spnLevels.Value = spnLevels.Max
  txtLevels.Text = Ebenen
  HeadingsAuswahl
  Me.Tag = Selection.Range.Start
```

```
  savEnv = Options.Pagination
  Options.Pagination = False
End Sub

Private Sub lbOutline_Click()
  Dim oRange As Range
  oDoc.Repaginate
  oDoc.Range(hd(lbOutline.ListIndex, 2), hd(lbOutline.ListIndex, 2)).Select
  Set oRange = Selection.Bookmarks("\HeadingLevel").Range
    While left(oRange.Text, 1) = Chr(12)
    oRange.SetRange oRange.Start + 1, oRange.End
  Wend
  While right(oRange.Text, 1) = Chr(12)
    oRange.SetRange oRange.Start, oRange.End - 1
  Wend
  S = oDoc.Range(oRange.Start, oRange.Start).Information(wdActiveEndPageNumber)
  P = oDoc.Range(oRange.End - 1, oRange.End - 1).Information(wdActiveEndPageNumber)
  If P < S Then P = S
  lblBereich.Caption = S & "-" & P
  oRange.Select
End Sub

Private Sub spnLevels_Change()
  If Me.Tag = "*INIT" Then Exit Sub
  txtLevels.Text = spnLevels.Value
  HeadingsAuswahl
End Sub

Private Sub cmdGeheZu_Click()
  Me.Tag = hd(lbOutline.ListIndex, 2)
  Unload Me
End Sub

Private Sub cmdDrucken1_Click()
  oDoc.PrintOut Range:=wdPrintRangeOfPages, _
     Pages:=lblBereich.Caption
End Sub

Private Sub cmdDrucken2_Click()
  With Dialogs(wdDialogFilePrint)
    .Range = wdPrintRangeOfPages
    .Pages = lblBereich.Caption
    .Show
  End With
  lblDrucker.Caption = ActivePrinter
End Sub

Private Sub cmdSchliessen_Click()
  If Me.Tag <> "*INIT" Then oDoc.Range(Me.Tag, Me.Tag).Select
  Options.Pagination = savEnv
  Unload Me
End Sub
```

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  cmdSchliessen_Click
End Sub

Private Function HeadingsSammeln() As Long
  Dim Absatz As Paragraph
  ReDim H(oDoc.Paragraphs.Count, 2) As String
  I = -1
  For Each Absatz In oDoc.Paragraphs
    If Absatz.OutlineLevel <= 9 Then
      I = I + 1
      tmp = left(Absatz.Range.Text, Len(Absatz.Range.Text) - 1)
      While left(tmp, 1) = Chr(12)
        tmp = Mid(tmp, 2)
      Wend
      H(I, 0) = LTrim(Absatz.Range.ListFormat.ListString & " " & tmp)
      H(I, 1) = Absatz.OutlineLevel
      H(I, 2) = Absatz.Range.Start
    End If
  Next
  If I = -1 Then
    MsgBox "Das Dokument enthält keine Überschriften.", vbInformation, Titel
    HeadingsSammeln = 4
    Exit Function
  End If
  ReDim ol(I, 2)
  For j = 0 To I
    ol(j, 0) = String((H(j, 1) - 1) * 2, " ") & H(j, 0)
    ol(j, 1) = H(j, 1)
    If H(j, 1) > Ebenen Then Ebenen = H(j, 1)
    ol(j, 2) = H(j, 2)
  Next j
  ReDim H(0, 0)
  HeadingsSammeln = 0
End Function

Private Sub HeadingsAuswahl()
  j = -1
  For I = 0 To UBound(ol)
    If CLng(ol(I, 1)) <= spnLevels.Value Then
      j = j + 1
    End If
  Next I
  If j = -1 Then
    lbOutline.Clear
    cmdGeheZu.Enabled = False
    cmdDrucken1.Enabled = False
    cmdDrucken2.Enabled = False
    lblBereich.Caption = ""
    Exit Sub
  End If
  If lbOutline.ListIndex > -1 Then oldpos = hd(lbOutline.ListIndex, 2)
  ReDim hd(j, 2)
```

```
  newpos = 0
  j = -1
  For I = 0 To UBound(ol)
    If CLng(ol(I, 1)) <= spnLevels.Value Then
      j = j + 1
      hd(j, 0) = ol(I, 0)
      hd(j, 1) = ol(I, 1)
      hd(j, 2) = ol(I, 2)
      If CLng(hd(j, 2)) <= oldpos Then newpos = j
    End If
  Next I
  lbOutline.List = hd()
  lbOutline.ListIndex = newpos
  cmdGeheZu.Enabled = True
  cmdDrucken1.Enabled = True
  cmdDrucken2.Enabled = True
End Sub

Private Sub SchaltflaechenErstellen()
  Set lbOutline = Controls.Add("Forms.ListBox.1", "lbOutline", True)
  With lbOutline
    .left = 6
    .top = 6
    .Width = 210
    .Height = 200
  End With
  Set frm = Controls.Add("Forms.Frame.1", , True)
  With frm
    .left = 225
    .top = 3
    .Width = 72
    .Height = 27
    .Caption = "Seitenbereich"
  End With
  Set lblBereich = frm.Controls.Add("Forms.Label.1", "lblBereich", True)
  With lblBereich
    .left = 6
    .top = 6
    .Width = 54
    .Height = 24
  End With
  Set frm = Controls.Add("Forms.Frame.1", , True)
  With frm
    .left = 225
    .top = 33
    .Width = 72
    .Height = 51
    .Caption = "Drucker"
  End With
  Set lblDrucker = frm.Controls.Add("Forms.Label.1", "lblDrucker", True)
  With lblDrucker
    .left = 6
    .top = 6
```

```
    .Width = 48
    .Height = 36
    .Caption = ActivePrinter
  End With
  Set frm = Controls.Add("Forms.Frame.1", , True)
  With frm
    .left = 225
    .top = 87
    .Width = 72
    .Height = 30
    .Caption = "Ebenen"
  End With
  Set txtLevels = frm.Controls.Add("Forms.TextBox.1", "txtLevels", True)
  With txtLevels
    .left = 36
    .top = 5
    .Width = 15
    .Height = 15
    .Enabled = False
    .TextAlign = fmTextAlignRight
  End With
  Set spnLevels = frm.Controls.Add("Forms.SpinButton.1", "spnLevels", True)
  With spnLevels
    .left = 54
    .top = 4
    .Width = 9
    .Height = 18
  End With
  oben = Me.Height - 42
  For I = 4 To 1 Step -1
    Set cmd = Controls.Add("Forms.CommandButton.1", , True)
    With cmd
      .left = 225
      .top = oben
      .Width = 72
      .Height = 18
      Select Case I
        Case 1
          .Caption = "GeheZu"
          .Accelerator = "G"
          .Default = True
          Set cmdGeheZu = cmd
        Case 2
          .Caption = "Direkt Drucken"
          .Accelerator = "D"
          Set cmdDrucken1 = cmd
        Case 3
          .Caption = "Indirekt Drucken"
          .Accelerator = "I"
          Set cmdDrucken2 = cmd
        Case 4
          .Caption = "Schliessen"
          .Accelerator = "S"
```

```
          .Cancel = True
          Set cmdSchliessen = cmd
      End Select
    End With
    oben = oben - 21
  Next I
End Sub
```

(Dieser Code muss in die UserForm mit dem Namen *frmKapitelDrucken*)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Kapiteld.htm

René Probst, November 2000

Änderungen

- 17. Januar 2001 - Verhindern, dass der Seitenumbruch im Hintergrund statt findet

Wie lassen sich Textmarken in einem Word-Dokument bequem ansteuern?

◄ Zurück zum Hauptindex

Frage

> In meinen Word-Dokumenten arbeite ich oft mit zahlreichen Textmarken. Der integrierte Word-Dialog *Einfügen - Textmarken* ist nach meiner Meinung aber keine effiziente Mimik um diese Textmarken schnell anzusteuern. Gibt es eine praktischere Alternative dazu?

Antwort

Im folgenden finden Sie zwei Tools, welche das effiziente Ansteuern von Textmarken in einem Dokument erlauben. Beide Lösungen arbeiten mit einem zusätzlichen Popup-Menü, welches Sie auf einer Symbolleiste Ihrer Wahl platzieren können und welches sämtliche Textmarken im Dokument auflistet. Mit einem einfachen Klick auf einen Textmarkenname in dieser Auflistung wird dann zur gewünschten Stelle im Dokument gesprungen.

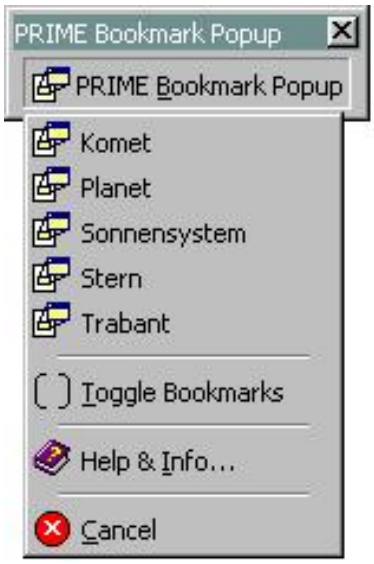
Das erste Tool können Sie von der Software-Firma mit dem Namen **PRIME** über das Internet herunter laden. Es installiert sich dann auf Knopfdruck und Sie dürfen diese Lösung frei benutzen. Das Tool nennt sich übrigens **PRIME Bookmark Popup**.

Eine Alternative dazu wird direkt hier vorgestellt. Wie sie diese Lösung einrichten, wird anschliessend Schritt für Schritt erklärt.

Die beiden Lösungen ähneln sich recht stark, wobei die hier vorgestellte Lösung einen etwas grösseren Funktionsumfang hergibt. Im folgenden wird die Funktionalität bzw. werden die Unterschiede der beiden Tools aufgelistet.

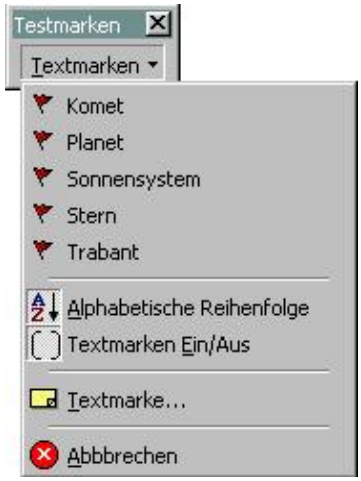
Funktioneller Vergleich der Lösungen

PRIME Bookmark Popup



- Nach dem Download können Sie das Tool auf Knopfdruck installieren.
- Das Projekt ist gesperrt. Sollten Sie Interesse haben, den VBA-Code dazu einzusehen, ist dies nicht möglich.
- Alle Textmarken im Dokument werden aufgelistet und Sie können eine bestimmte Textmarke mit einem einfachen Klick anspringen.
- Die Textmarken erscheinen in der Auflistung in alphabetischer Reihenfolge.

Meine Lösung



- Das Tool lässt sich nicht auf Knopfdruck installieren. Um die Lösung verfügbar zu machen, gehen Sie nach der Anweisung im nächsten Kapitel vor.
- Der VBA-Code ist einsehbar, er folgt nämlich gleich anschliessend in diesem Skript.
- Alle Textmarken im Dokument werden aufgelistet und sie können eine bestimmte Textmarke mit einem einfachen Klick anspringen (in diesem Punkt also identisch mit der ersten Lösung).
- Die Textmarken erscheinen in der Auflistung **wahlweise** in alphabetischer Reihenfolge oder in der Reihenfolge, wie diese im Text erscheinen.

Das Textmarken-Utility

Im folgenden finden Sie meinen Lösungsvorschlag um den Umgang mit Textmarken in einem Word-Dokument zu erleichtern.

Implementierung der Lösung

Um diese Lösung auf Ihrem Rechner verfügbar zu machen, gehen Sie bitte wie folgt vor:

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** (über Datei - Öffnen)
2. Wechseln Sie mit der Tastenkombination **[Strg + F11]** in die VBA-Umgebung.
3. Wählen Sie im VBA-Editor im Hauptmenü den Dialog ***Einfügen - Modul*** und kopieren Sie den unten abgebildete VBA-Code in das leere Code-Fenster.
4. Kehren Sie in das Dokumentfenster zurück.
5. Rufen sie vom Word-Hauptmenü den Dialog ***Extras - Makro - Makros...*** auf und lassen Sie das Makro mit dem Namen **TextmarkenMenüEinrichten ausführen**. Diese Aktion muss genau einmal geschehen. Die Schaltfläche wird damit in der Standardvorlage mit den Namen **Normal.dot** erstellt und bleibt dort gespeichert.
6. Wählen Sie vom Hauptmenü den Dialog ***Extras - Anpassen...***.
7. Aktivieren Sie den Reiter **Befehle**.
8. Markieren Sie in der linken Liste mit der Bezeichnung *Kategorien:* den Eintrag **Makros**.
9. Suchen Sie in der rechten Liste mit der Bezeichnung *Befehle* den Eintrag **TextmarkenSammeln**.
10. Ziehen Sie diesen Eintrag mit der Maus auf eine Symbolleiste Ihrer Wahl und legen Sie die Schaltfläche dort ab.
11. Optional können Sie immer noch in diesem Anpassungsmodus einen Rechtsklick auf die Schaltfläche ausüben, und über das Kontextmenü Name und Aussehen der Schaltfläche beeinflussen.
12. Speichern Sie nun die *Normal.dot* und verlassen Sie Word.
13. Nachdem Sie Word wieder gestartet habe, können Sie die neue Funktion nutzen.

VBA-Code

Finden Sie im Folgenden den Code, welchen Sie in ein neues Modul der Standardvorlage mit dem Namen **Normal.dot** kopieren sollten.

```
Private Const Leiste = "Standard"
Private Const Tag = "SW19991006"

Sub TextmarkenMenüEinrichten()
  CustomizationContext = NormalTemplate
  Dim Menü As CommandBarPopup
  Set Menü = CommandBars.FindControl(Type:=msoControlPopup, Tag:=Tag)
  If Not Menü Is Nothing Then Menü.Delete
  Set Menü = CommandBars(Leiste).Controls.Add(Type:=msoControlPopup)
  Dim Knopf As CommandBarButton
  With Menü
    .Tag = Tag
    .Caption = "&Textmarken"
    .BeginGroup = True
    .TooltipText = "Zeigt alle Textmarken im Dokument zur Auswahl an"
    .OnAction = "TextmarkenSammeln"
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .Tag = Tag & "1"
      .BeginGroup = True
      .Caption = "&Alphabetische Reihenfolge"
      .FaceId = 928
      .OnAction = "TextmarkenSortierung"
    End With
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .Tag = Tag & "2"
      .Caption = "Textmarken &Ein/Aus"
      .FaceId = 2636
      .OnAction = "TextmarkenToggle"
    End With
    Set Knopf = .Controls.Add(id:=758)
    With Knopf
      .BeginGroup = True
      .FaceId = 1546
    End With
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .BeginGroup = True
      .Caption = "&Abbbrechen"
      .FaceId = 1019
    End With
  End With
  NormalTemplate.Save
End Sub

Private Sub TextmarkenSammeln()
  savSettings1 = NormalTemplate.Saved
```

```
savSettings2 = ActiveDocument.AttachedTemplate.Saved
Dim Menü As CommandBarPopup, Knopf As CommandBarButton
Set Menü = CommandBars.ActionControl
For Each c In Menü.Controls
  If c.Tag = Tag & "3" Or c.Tag = Tag & "4" Then c.Delete
Next
AnzTM = ActiveDocument.Bookmarks.Count
If AnzTM = 0 Then
  Set Knopf = Menü.Controls.Add(Type:=msoControlButton, Before:=1)
  With Knopf
    .Tag = Tag & "4"
    .BeginGroup = True
    .Caption = "Keine Textmarken"
    .FaceId = 1786
  End With
Else
  Dim oDoc As Document, x() As String, y() As String
  ReDim x(AnzTM - 1)
  ReDim y(AnzTM - 1, 1)
  Set oDoc = ActiveDocument
  For i = 0 To AnzTM - 1
    x(i) = oDoc.Bookmarks(i + 1).Name
    y(i, 1) = x(i)
    If oDoc.Bookmarks(i + 1).Range.StoryType = wdMainTextStory Then
      y(i, 0) = Format(oDoc.Bookmarks(i + 1).Range.Start, "00000000")
    Else
      y(i, 0) = 1000000000
    End If
  Next i
  If oDoc.Bookmarks.DefaultSorting = wdSortByLocation Then
    If AnzTM > 1 Then WordBasic.SortArray y()
  End If
  For i = 0 To AnzTM - 1
    Set Knopf = Menü.Controls.Add(Type:=msoControlButton, Before:=i + 1)
    With Knopf
      If oDoc.Bookmarks.DefaultSorting = wdSortByName Then
        .Caption = x(i)
      Else
      End If
      .Caption = y(i, 1)
      .Tag = Tag & "4"
      .FaceId = 1678
      .OnAction = "TextmarkenSprung"
    End With
  Next i
End If
Set Knopf = Menü.CommandBar.FindControl(Tag:=Tag & "1")
If Not Knopf Is Nothing Then
  If ActiveDocument.Bookmarks.DefaultSorting = wdSortByName Then
    Knopf.State = msoButtonDown
  Else
    Knopf.State = msoButtonUp
  End If
```

```
  End If
  Set Knopf = Menü.CommandBar.FindControl(Tag:=Tag & "2")
  If Not Knopf Is Nothing Then
    If ActiveWindow.View.ShowBookmarks = True Then
      Knopf.State = msoButtonDown
    Else
      Knopf.State = msoButtonUp
    End If
  End If
  NormalTemplate.Saved = savSettings1
  ActiveDocument.AttachedTemplate.Saved = savSettings2
End Sub

Private Sub TextmarkenSprung()
  ActiveDocument.Bookmarks(CommandBars.ActionControl.Caption).Select
End Sub

Private Sub TextmarkenToggle()
  ActiveDocument.ActiveWindow.View.ShowBookmarks = _
     Not ActiveDocument.ActiveWindow.View.ShowBookmarks
End Sub

Private Sub TextmarkenSortierung()
  If CommandBars.ActionControl.State = msoButtonUp Then
    ActiveDocument.Bookmarks.DefaultSorting = wdSortByName
    CommandBars.ActionControl.State = msoButtonDown
  Else
    ActiveDocument.Bookmarks.DefaultSorting = wdSortByLocation
    CommandBars.ActionControl.State = msoButtonUp
  End If
End Sub
```

(Das praktische Textmarken-Utility)

Links zu diesem Thema

PRIME Software Products

► **PRIME Bookmark Popup**

Eigene Links

► **Symbolleistenschaltflächen und Menüleisten selbst gemacht**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/TMPopup.htm

René Probst, Januar 2001

Änderungen

- --- nil ---

MS Word und Schriften

◄ Zurück zum Hauptindex

Problem

1. Nachdem ich eine neu erworbene Schrift in den Schriftenordner kopiert habe, steht diese Schrift zwar in Excel zu Verfügung nicht aber unter MS Word.

2. Ich bette Schriften direkt in das Word-Dokument ein. Trotzdem, wenn ich das Dokument weitergebe, wird der Text beim Empfänger nicht mit den von mir vorgesehen Schriften angezeigt.

3. Nachdem ich auf eine höhere Version von MS Office umgestiegen bin, steht die Schrift mit dem Namen *MS LineDraw* zwar noch unter Excel nicht aber unter MS Word zur Verfügung.

Diese Seite

▼ Ohne grafikfähigen Druckertreiber läuft gar nichts!
▼ Zusätzliche Schriften ordnungsgemäss installieren
▼ Wie viele Schriften darf ich maximal unter Windows installieren?
▼ Schriften im Word-Dokument einbetten
▼ Schriften, welche von MS Word nicht mehr unterstützt werden
▼ Zusätzliche Werkzeuge von Microsoft für den Schriftenordner
▼ Proportionale versus proportionale Schriften
▼ Symbolschriften
▼ Zusätzliche Schriften von Microsoft, welche das Euro-Symbol unterstützen
▼ Links zu diesem Thema

Ohne grafikfähigen Druckertreiber läuft gar nichts!

▲ Zurück zum Anfang

MS Word arbeitet sehr eng mit dem jeweils aktiven Druckertreiber zusammen. Damit unterscheidet sich MS Word von vielen anderen Anwendungen. Dies bedingt, dass ein brauchbarer Druckertreiber auf Ihrem Rechner installiert ist. Unter brauchbar ist vor allem gemeint, er muss grafikfähig sein. Gut beraten sind Sie auch, wenn Sie sich eine möglichst aktuelle Version des Treibers besorgen. Das gilt um so mehr, wenn Sie sehr moderne Anwendungssoftware einsetzten oder im Begriff sind darauf umzustellen. Sie müssen auch dann einen Drucker installieren, wenn an Ihrem Rechner physisch gar kein Drucker angeschlossen ist. Missachten Sie dies, dann wird MS Word nicht einmal mehr fähig sein, die Schriften anzuzeigen und Ihnen zur Disposition zu stellen. Auch die Seitenlayout- und die Seitenansicht kann dann nicht richtig funktionieren. Für Sie heisst dies:

- ! Installieren Sie einen **grafikfähigen** Drucker.
- ! Küren Sie diesen Drucker unter Windows zum **Standarddrucker**.
- ! Achten Sie darauf, dass der Druckertreiber **nicht älter als die Anwendungssoftware** ist, welche Sie einsetzen.

Mehr über dieses Thema erfahren Sie im folgenden Skript:

► Es stehen keine oder nur zwei Schriften zur Verfügung

Zusätzliche Schriften ordnungsgemäss installieren

▲ Zurück zum Anfang

Microsoft stellt eine sinnvolle Auswahl von verschiedenen Schriftarten zur Verfügung. Einige dieser Schriften werden installiert, wenn Sie Windows aufsetzen. Auch mit der Installation des MS Office-Paketes kommen zusätzliche Schriften hinzu insbesondere die Schrift mit dem Namen **Tahoma**, welche Voraussetzung ist, dass gewisse Office-Dialoge schön und gut lesbar angezeigt werden können.

Alle häufige verwendeten Schriftarten, sind sogenannte **True Type Fonts**. Bei dieser Technologie liegen die Zeichen der Schrift nicht als Punkteraster vor, sondern werden durch **Vektoren** beschrieben. Das heisst, sie liegen nicht bildlich vor, es ist viel mehr das Kochrezept dazu vorhanden. Damit sind die Zeichen auf mannigfache Art und Weise skalierbar (vergrössern, dehnen, stauchen) etc. ohne dass z.B. bei der Vergrösserung die Qualität (Auflösung) schlechter würde. Das heisst also, es liegt intern kein *Lupeneffekt* vor. So wie Sie etwas ausdrucken unterliegt dies selbstverständlich den Gesetzten der Materie. Dateien, welche eine Schriftart darstellen weisen die Dateierweiterung **TTF** auf.

Auch was die Installation von Schriften betrifft, ist MS Word etwas anspruchsvoller als andere Anwendungen, wie z.B. Excel. Es reicht nämlich nicht aus, eine neue Schriftdatei in den Schriftenordner zu kopieren. MS Word wird diese zusätzliche Schriftart nicht in jedem Falle anzeigen, da diese im technischen Sinne nicht registriert ist, also nicht in der Registrierdatenbank eingetragen ist. Um eine zusätzliche Schrift ordnungsgemäss zu installieren, gehen Sie wie folgt vor:

1. Halten Sie den Datenträger mit der TTF-Datei bereit.
2. Unter Windows wählen Sie ***Einstellungen - Systemsteuerung.***
3. Führen Sie einen Doppelklick auf das Symbol **Schriftarten** aus.
4. Wenn der Schriftenordner angezeigt wird, dann wählen Sie dort vom Menü den Befehl ***Datei - Neue Schriftart installieren...***. Sollte es in Ihrem Schriftenordner kein solchen Befehl im Menü *Datei* geben, dann ist der Schriftenordner angeschlagen (defekt) und muss repariert werden. Lesen Sie dazu das Kapitel *Zusätzliche Werkzeuge von Microsoft für den Schriftenordner*
5. Es wird ein Dialog angezeigt, welcher es Ihnen erlaubt, die gewünschte TTF-Schrift auf Ihrem Datenträger oder in einem Verzeichnis anzuwählen.
6. Schliessen Sie nun die Installation mit **Ok** ab.

Wie viele Schriften darf ich maximal unter Windows installieren?

▲ Zurück zum Anfang

Eingangs möchte ich erwähnen, dass Sie sich was die Anzahl der Schriften betrifft, auf ein vernünftiges und adäquates Quantum von Schriften beschränken sollten. Schauen Sie sich dazu den Schriftenordner an, welchen Sie bequem über ***Start - Einstellungen - Systemsteuerung - Schriftarten*** erreichen können. Wenn in diesem (speziellen) Ordner die Gesamtzahl der Dateien zwischen 300 - 600 liegt, dann ist das ein vernünftiges Mass. Liegt die Zahl deutlich über 600 sind zu mindest unter Windows 9x Probleme zu erwarten. Die Datenbank von Microsoft dokumentiert in diesem Falle z.B. Probleme, im Bereich MS Formeleditor und MS Organigramm.

Was immer Sie auch für ein Betriebssystem im Einsatz haben, bedenken Sie, dass Schriften Arbeitsspeicher belegen. Dabei handelt es sich um residenten Arbeitsspeicher, also Speicher, welcher nicht ausgelagert werden kann.

Zumindest unter Windows 9x ist die Anzahl der Schriften, welche Sie installieren können auch in der Praxis klar begrenzt. Diese Begrenzung erreichen Sie dann, wenn sich im Schriftenordner ca.1000 Schriften tummeln (tendenziell etwas mehr). Die obere Grenze kann exakt so nicht angegeben werden. Es hängt auch von der Länge der Namen der Schriftdateien ab. Bei rund 1000 beginnt aber ein Buffer (Zwischenspeicher), welcher Windows für das Verwalten von Schriften benötigt zu überlaufen.

Dummerweise ist Windows nicht sehr kommunikativ in diesem Bereich. Sie werden einfach feststellen, dass der Schriftenordner keine weiteren Schriften mehr aufnimmt. Windows wird Ihnen das nicht mitteilen sondern sich stumm verweigern. Wir können also folgendes festhalten:

! (Zu viele) Schriften vergeuden wertvollen Arbeitsspeicher. Sind noch weitere Speicherfresser auf der Gasse, kann dies die Performanz und das zuverlässige Funktionieren Ihres Rechners beinträchtigen. Mehr als gut **1000** Schriftdateien (wobei eine Schriftart aus mehreren Schriftdateien bestehen kann) sind unter **Windows 9x** nicht möglich. Diese werden schlicht nicht mehr angenommen.

Schriften im Word-Dokument einbetten

▲ Zurück zum Anfang

Wenn Sie ein Word-Dokument als Datei weiter geben, wird in der Regel stillschweigend angenommen, dass die in diesem Dokument verwendeten Schriftarten auch auf dem Rechner des Empfängers installiert sind. Wie man sich leicht denken kann, ist diese Voraussetzung aber in der Praxis nicht immer gegeben.

Sollte eine bestimmte Schrift, welche im Dokument Verwendung findet auf dem Rechner nicht (mehr) vorhanden sein, dann tritt - vom Rechner ausgelöst - *Plan B* in Aktion und der Text wird in einer Schriftart angezeigt, welche auf dem Rechner tatsächlich installiert ist und der ursprünglichen Schrift möglichst ähnlich ist. Dieser sehr sinnvolle Hokuspokus kann dadurch erreicht werden, dass Schriften in Tat und Wahrheit in Kategorien eingeteilt sind und zwar gleich auf verschiedenen Ebenen. Diese Vorkehrung ermöglicht es dem Betriebssystem an Hand einer Art Rastersuche, was adäquates aufzutreiben. Es wird tendenziell also nicht geschehen, dass eine sehr breite Schrift mit einer sehr gedungenen Schrift substituiert (also ersetzt) wird. Die ist um so wichtiger, weil die meisten Schriften, welche zum Einsatz kommen *proportionale Schriften* sind. Das ersetzen einer weiten Schrift durch eine enge Schrift, könnte also das Layout Ihres Dokumentes gehörig durcheinander bringen.

Um ganz sicher zu sein, dass der Text auch in der gewünschten Schrift erscheint und das Dokument möglichst layoutgetreu weiter gegeben werden kann, ziehen es manche Benutzer vor, die Schriftarten mit in das Dokument einzubetten. MS Word unterstützt diese Methode. Sie finden diese Option vom Word-Hauptmenü aus gesehen unter ***Extras - Optionen - Speichern - True Type Schriften einbetten***. Bedenken Sie aber, dass das Einbetten von Schriften die Dateigrösse sehr markant erhöhen kann. Nicht selten legt die Dateigrösse dabei um das 5-10 Fache zu. Der Prozess lässt sich allerdings problemlos umkehren.

Wenn MS Word auch dass Einbetten von Schriften unterstützt, sollten Sie sich im klaren sein, dass dieses Unterfangen in vielen Fällen fehlschlägt. Eine Fehlermeldung können Sie leider beim Fehlschlagen nicht erwarten, denn Word bleibt stumm. Sie merken es u.a. daran, dass die Dateigrösse nicht zunimmt. Dies sollte aber in sehr markanter Weise geschehen. Der Grund, dass viele Schriftarten gar nicht eingebettet werden können, liegt daran, dass entweder lizenztechnische Gegebenheiten dies verwehren (viele Schriften bzw. Schriftarten sind nämlich urheberrechtlich geschützt), oder dass die Schriftart nicht die technischen Schnittstellen dafür bietet, welche das Einbetten generell ermöglichen würde. Jemand, welche sich daran macht, eine TTF-Schrift zu kreieren, müsste nämlich, wenn er es perfekt machen würde, eine Unzahl von Schnittstellen vorsehen und eine ganze Liste von Schnittstellendefinitionen abarbeiten. Von diesem Unterfangen schrecken allerdings viele Entwickler zurück. Von den frei verfügbaren Schriften, welche ich kenne und die nicht von Herstellern wie z.B. Adobe oder Microsoft stammen, ist jedenfalls kaum eine olympiareif.

Wenn Sie es genau wissen möchten, was eine bestimmte Schrift kann und was diese nicht kann, dann laden Sie sich das Zusatztool von Microsoft herunter, welches in diesem Skript vorgestellt wird und installieren Sie dieses. Grundsätzlich gibt es folgende Stufen, was das Einbetten von Schriften angeht:

- Das Einbetten ist erlaubt, die Schrift darf permanent auf dem Rechner gespeichert werden.
- Das Einbetten ist erlaubt, die Schrift wird allerdings nur flüchtig auf dem Rechner geladen.
- Das Einbetten ist erlaubt, die Schrift wird allerdings nur flüchtig auf dem Rechner geladen und das Dokument darf ausschliesslich Lesezugriff erlauben.
- Das Einbetten ist nicht erlaubt.

Wir fassen also zusammen:

! Fehlende Schriften können bei der Weitergabe eines Dokumentes, dazu führen, dass das Dokument nicht layoutgetreu angezeigt oder gedruckt wird.

! Ersparen Sie sich selbst und dem Empfänger eines Dokumentes Ärger, in dem Sie ausschliesslich sehr verbreitet und frei verfügbare Schriften verwenden.

! Das Einbetten von Schriften schlägt ganz gewaltig auf die Dateigrösse.

! Viele Schriften sind urheberrechtlich geschützt.

! Viele Schriften dürfen nicht eingebettet werden bzw. lassen sich nicht Einbetten. Word wird das Misslingen beim Einbetten nicht monieren, es passiert einfach gar nichts.

! Achten Sie beim Kauf von Schriften auf Qualität und zwar nicht nur im typografischen Sinne, sondern auch im technischen Sinne. Qualität ist gegeben bei grossen Anbieter wie *Microsoft*, *Adobe* und *Corel* und sicher auch bei vielen kleineren Unternehmen. Qualität ist in der Regel nicht gegeben, wenn Sie für DM 19,99 1000 Schriften von einem anonymen Vertreiber erwerben. Oft fehlen dann die Umlaute, französische Zeichen mit Akzenten, etc. Das Euro-Symbol ist wieder eine Klasse für sich; sie finden es leider nur in wenigen Schriften. Gute zusätzliche Schriftensammlungen kosten typischerweise zwischen DM 80,00 und DM 250,00.

☛ Manchmal ist es fast preiswerter eine Software zu erwerben, welche u.a. gute Schriften beinhaltet, als ein Schriftenpaket zu kaufen. Ältere, aber neuwertige Programme von *Corel* z.B. finden Sie sehr günstig auf dem Markt.

☛ Sie finden am Schluss dieses Skriptes einen Link zu Microsoft, welcher es Ihnen erlaubt, zusätzliche Schriften, welche das Euro-Symbol enthalten, herunter zu laden.

Schriften, welche von MS Word nicht mehr unterstützt werden

▲ Zurück zum Anfang

MS Word unterstützt nicht alle Schriftarten. Microsoft hat auch gewissen Schriften, welche von früheren Versionen von MS Word noch unterstützt wurden zurückgezogen. Sie finden ganz am Ende dieses Skriptes einen Link zu einem Artikel von Microsoft, welcher eine Liste beinhaltet mit Schriften, welche unter MS Word nicht mehr zur Disposition stehen. Darunter fällt auch die recht populäre Schriftart mit dem Namen **MS LineDraw**. Diese Schrift wird durch eine andere Schrift mit der Bezeichnung *Courier New* ersetzt.

Zusätzliche Werkzeuge von Microsoft für den Schriftenordner

▲ Zurück zum Anfang

Microsoft stellt zwei zusätzliche Werkzeuge für den Schriftenordner zur Verfügung. Dabei handelt es sich einerseits um ein Tool, welches einen defekten Schriftenordner reparieren kann. Dank einem weiteren Tool können Sie einer bestimmten Schrift direkt vom Schriftenordner aus bequem über das Kontextmenü zahlreiche ergänzende Informationen entlocken.

Werkzeug um den Schriftenordner zu reparieren (*Tweak UI*)

Beim Schriftenordner handelt es sich um einen magischen Ordner. Wird dieser Ordner nämlich im Windows-Explorer aufgerufen, dann überrascht er durch zusätzliche Funktionen, welche andere Ordner nicht kennen. Sollte dies nicht der Fall sein, was leider recht häufig der Fall ist, dann hat das nicht besonders stabile Windows einen Defekt in diesem Bereich. In diesem Falle muss der Schriftenordner repariert werden, weil es insbesondere nicht mehr möglich ist, Schriften ordnungsgemäss zu installieren.

Die *Azubis* von Microsoft haben ein Tool geschaffen, um gewisse zusätzliche Einstellungen unter Windows zu vollführen, an welche Sie sonst schlecht ran kommen, bzw. gar nicht, wollen Sie es vermeiden, direkt in die Registrierdatenbank einzugreifen. Das Werkzeug nennt sich **Tweak UI** und nistest sich als zusätzliches Symbol in der Systemsteuerung von Windows ein. Den Link für den Download finden Sie am Schluss dieses Skriptes. Vollziehen Sie folgende Schritte, um das Werkzeug auf Ihrem Rechner verfügbar zu machen.

Tweak UI herunter laden und installieren

1. Laden Sie vom Microsoft-Server das Werkzeug mit dem Namen *Tweak UI* herunter. Die Download-Adresse finden Sie am Schluss dieses Skriptes.
2. Entpacken Sie die heruntergeladene Datei.
3. Führen Sie einen Rechtsklick auf die extrahierte Datei mit dem Namen **tweakui.ini** aus und wählen Sie im Kontextmenü **Installieren**.
4. Folgen Sie den Installationsanweisung und führen Sie die Installation zu Ende.

Schriftenordner mit Tweak UI reparieren

1. Klicken Sie unter Windows ***Start - Einstellungen - Systemsteuerung*** an.
2. Sie sollten in der Systemsteuerung eine neues Symbol vorfinden mit der Bezeichnung **Tweak UI**.
3. Doppelklicken Sie dieses Symbol und wechseln Sie im Folgedialog in den Reiter **Repair**.
4. Im Kombinationsfeld stellen Sie nun den Eintrag **Repair Font Folder** ein und klicken anschliessend auf den Schaltknopf **Repair Now**.
5. Starten Sie Ihren Rechner neu. Der Schriftenordner, wird nun wieder über die zusätzliche Funktionalität verfügen, welche z.B. nötig ist, um Schriften ordnungsgemäss zu installieren.

Anmerkung

- Sollten Sie darauf Wert legen, dieses Werkzeug mit einer deutschsprachigen Benutzeroberfläche zu erhalten, verweise ich Sie auf die Windows 98 CD oder auf die zahlreichen in Deutschland erscheinenden PC-Magazine, denen dieses Tool in unregelmässigen Abständen als Zugabe auf CD-ROM vorliegt. Man berichtet mir, dass *Tweak UI* auf der **Windows 98 2nd Edition** nicht mehr zu finden ist. Bei der ersten Ausgabe ist das noch der Fall. Wie auch immer, achten Sie, dass das Werkzeug, welches Sie installieren mit Ihrem Betriebssystem vereinbar ist.

Zusatzwerkzeug von Microsoft um einer Schriftart zusätzliche Informationen zu entlocken

Wenn Sie es genau wissen wollen, z.B. ob eine bestimmte Schriftart die Voraussetzungen erfüllt, eingebettet zu werden und in welchem Umfang dies möglich ist, dann sollten Sie sich ein sehr schönes zusätzliche Werkzeug von Microsoft beschaffen. Der Eigenschaftsdialog, welcher angezeigt wird, wenn Sie im Schriftenordner einen Rechtsklick auf eine bestimmte Schriftart ausführen, wird durch dieses Tool stark erweitert.

Dieses Tool nennt sich **Font properties extension, version 2.1**. Die Adresse für den Download finden Sie ganz am Ende dieses Skriptes.

Proportionale versus proportionale Schriften

▲ Zurück zum Anfang

Im Zusammenhang mit der tabellarischen Darstellung von Text und beim Arbeiten mit Formularen kommt immer mal wieder die Frage auf, wie man es schafft, dass 10 Zeichen wirklich 10 Zeichen sind. Damit meine ich, dass Wörter, welche aus 10 Zeichen bestehen unabhängig davon, welche Zeichen die Wörter enthalten, immer genau die selbe Breite, also gleich viel Platz einnehmen.

Zur Zeit von mechanischen Schreibmaschinen, war diese Frage überhaupt kein Thema, konnten diese Geräte doch ausschliesslich Text mit nicht proportionalen Zeichen zu Papier bringen. Das heisst gleichgültig was für ein Zeichen, gleichgültig ob ein "i" oder ein "w", das Zeichen nahm immer genau den selben Platz in der Breite ein. Schriftarten, welche auf diesem Prinzip beruhen, nennen sich **nicht proportionale** Schriften oder auf Grund ihres Verwendungszweckes auch Schreibmaschinenschriften.

Textverarbeitungsprogramme auf Rechnern, verwenden vorwiegend nur noch proportionale Schriften, wie wir das ja auch vom Buch- und Zeitungsdruck her kennen. Diese Schriften geben ein wesentlich schöneres Schriftbild ab, und werden vom Leser als angenehmer empfunden. Sollten Sie doch mal auf Schreibmaschinenschriften zurückgreifen müssen, dann verwenden Sie am besten die Schrift mit dem Namen **Courier New**. Bei Schreibmaschinenschriften wirken breite Buchstaben unnatürlich gestaucht und schmale Buchstaben gestreckt. Schreibmaschinenschriften benötigen übrigens im Durchschnitt mehr Platz bei einem gleichlangen Text, als proportionale Schriften. Im folgenden Beispiel können Sie den Unterschied zwischen proportionalen Schriften und Schreibmaschinenschriften deutlich erkennen.

W.i.e.s.o

W.i.e.s.o

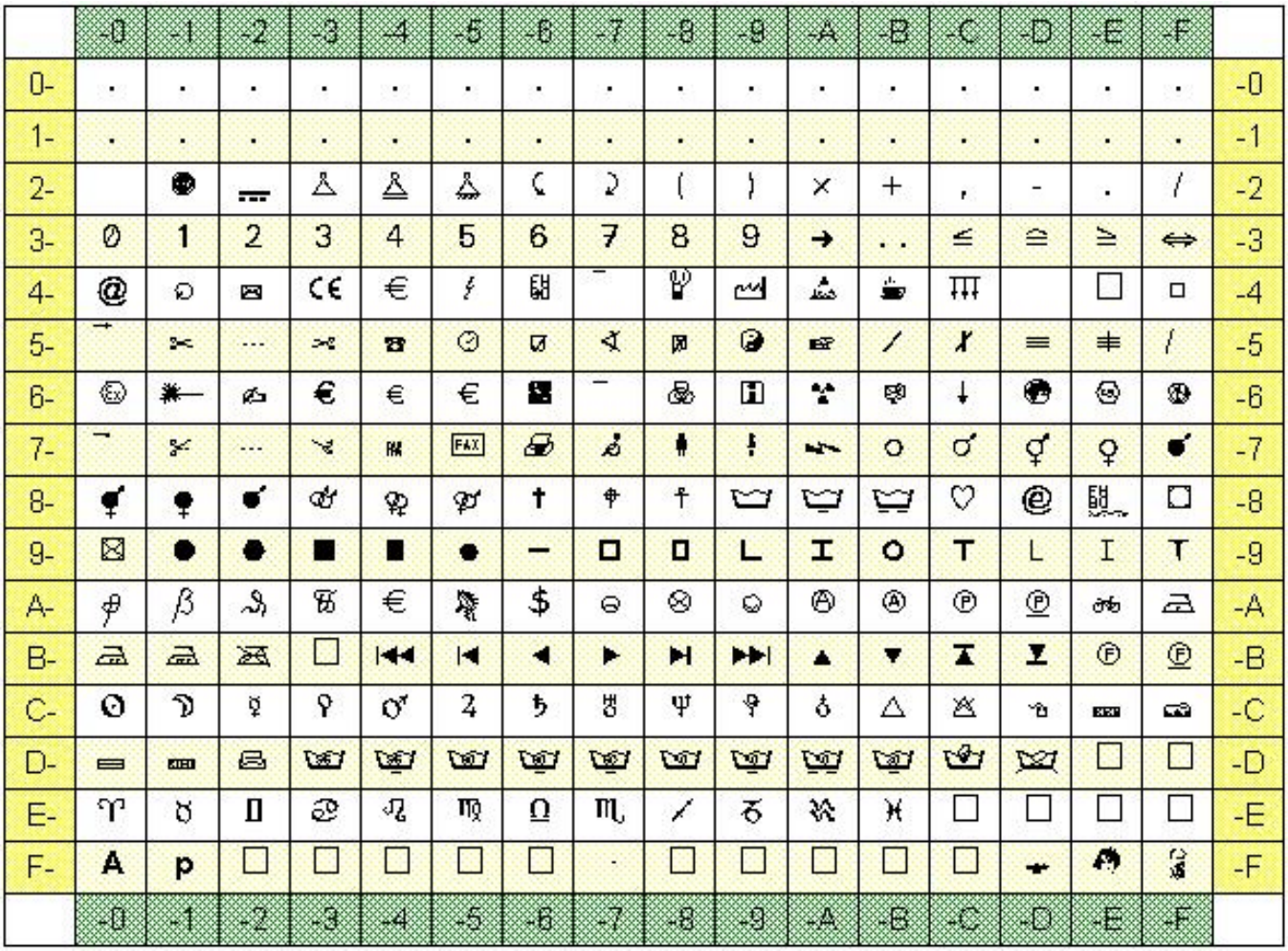
Symbolschriften

▲ Zurück zum Anfang

So verschieden Schriften auch aussehen mögen, die meisten davon beachten den ANSI-Standard ganz genau. ANSI (*American Standard Code for Information Interchange*) legt fest, welches Zeichen, welcher Position in der Zeichentabelle entsprechen soll.. Der Buchstabe "A" mag ja in zwei verschiedenen Schriften ganz individuell daher kommen, es ist und bleibt aber ein "A". Es gibt aber auch (sinnvolle) Ausnahmen . Dazu gehören sicher die diversen Symbolschriften, wie z.B. *Wingdings* oder *Webdings*. Diese Schriften kümmern sich keinen Deut um den ANSI-Standard, sondern zeigen keck beim betätigen der Taste "A" ein Smily-Symbol oder ähnliches an.

Wenn sie unter MS Word den Dialog ***Einfügen - Symbol*** bzw. je nach Word-Version den Dialog ***Einfügen - Sonderzeichen*** aufrufen, dann finden Sie eine engere Wahl von Schriftarten, welche Symbolschriften darstellen. Es ist nicht möglich, eine Schrift forciert in diesen Dialog zu laden. Ob es sich bei eine Schrift um eine Symbolschrift handelt,, muss in der Kopfdefinition der jeweiligen Schrift festgelegt sein. Über dieses Potential verfügt aber nur derjenige, welche die Schrift kreiert und dann auch im technischen Sinne als Schriftdatei verpackt.

Eine zusätzliche Symbolschrift mit sehr nützlichen Symbolen, welche sie herunter laden und frei benützen können stammt von Martin Vogel. Der direkte Link um diese Symbolschrift herunter zu laden, finden Sie am Ende dieses Skriptes. Die Zeichentabelle für diese Schrift (bzw. die Symboltabelle) finden Sie in der folgenden Abbildung.



Zusätzliche Schriften von Microsoft, welche das Euro-Symbol unterstützen

▲ Zurück zum Anfang

Damit eine Schrift das Euro-Währungssymbol anzeigen kann, muss die Schriftart dieses Zeichen unterstützen. Bei ganz populären Schriften, wie *Arial* und *Times New Roman* ist das meistens der Fall. Sollte dass bei Ihnen nicht klappen dann überprüfen Sie folgendes:

- Arbeiten Sie noch mit **Windows 95** oder einer älteren Version von **Windows NT**, dann müssten Sie unter Windows zuerst den Euro-Update einspielen. Den Link dazu finden Sie ganz am Ende dieses Skriptes.
- Sind auf Ihrem Rechner noch die alten Versionen der Schriften *Arial* und *Times New Roman* installiert? Sollte dies der Fall sein, dann besorgen Sie sich vom Microsoft-Server die jeweils neue Version dieser und anderer Schriften. Auch diesen Link finden Sie am Schluss dieses Skriptes.
- Wird das Euro-Symbol zwar angezeigt, aber nicht gedruckt? - In diesem Falle sollten Sie sich um einen neuen Druckertreiber bemühen. Beachten Sie dafür die *Home Page* Ihres Druckerherstellers.

Das Euro-Symbol geben Sie übrigens mit der Tastenkombination **[AltGr + E]** ein. Wie schon angedeutet, finden Sie am Schluss dieses Skriptes eine Adresse, von welcher Sie gut ein Duzend bekannter Schriften herunter laden können und zwar in einer neuen Version, also mit Euro-Symbol.

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Es stehen keine oder nur zwei Schriften zur Verfügung
► Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen?

Microsoft

Deutsch

► Weitere eurofähige Schriftarten für Microsoft Office
► Update: Unterstützung für das Euro-Währungssymbol in Windows 95

Englisch

► Tweak UI 1.33
► Font properties extension, version 2.1
► TrueType core fonts for the Web (Unicode-fähige Schriften von Microsoft)

Martin Vogel

► Der Symbolfont MARVOSYM.TTF - Nützliche Symbole für den Büroalltag

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Schrift.htm

René, Probst, Februar 2001

Änderungen

- --- nil ---

Wissenswertes über die Zeichensatztabellensuppe

◄ Zurück zum Hauptindex

Diese Seite

▼ Von Bit und Byte
▼ Der alte ASCII 7-Bit Code
▼ Der erweiterte ASCII-Code, EBCDIC und ANSI (8-Bit-Code)
▼ Standard- und Symbolschriften
▼ Die Schichten («Layers») im Zusammenhang mit Zeichentabellen
▼ Unicode oder «Universal Character Set» (UCS)
▼ Offene Probleme im Zusammenhang mit dem Unicode
▼ Unicode Schriftart installieren
▼ Wie erfahre ich den Code eines Unicode-Zeichens?
▼ Links zu diesem Thema

Von Bit und Byte

▲ Zurück zum Anfang

Ihr Rechner kann in letzter Konsequenz nur zwei Zustände unterscheiden, nämlich **0** und **1**. Diese kleinste denkbare Informationseinheit nennt sich **Bit**. Genau gleich, wie Sie mit den Ziffern 0-9 im Dezimalsystem nur zehn Zustände ausdrücken können, kennt das Binärsystem (auch Dualsystem genannt) mit den Ziffern 0 und 1 nur zwei Zustände. Anders, wenn Sie im Dezimalsystem zwei Ziffern gruppieren. Nun können Sie bereits 100 Zustände darstellen, bei drei Ziffern sind es bereits 10^3 also 1000 Zustände Ein typischer Zeichensatz kennt 128 - 256 Zeichen. Um all diese Zeichen abzubilden zu können, müssen also mehrere Bits zu einem sogenannten **Byte** gruppiert werden.

Der alte ASCII 7-Bit Code

▲ Zurück zum Anfang

Ursprünglich war ein **7-Bit-Code** gebräuchlich, welcher 2^7 Zeichen (das sind **128**) abbilden konnte. Dieser Code war bereits standarisiert und hiess **ASCII-Code** (*American Standard Code for Information Interchange*). Das reichte im angelsächsischen Raum für sämtliche Steuerzeichen (darunter versteht man Zeilenumbrüche, Tab-Steuerzeichen etc.), Grossbuchstaben, Kleinbuchstaben, Zahlen und die gebräuchlichsten Sonderzeichen wie Komma oder Punkt. Sehr bald und mit zunehmender Internationalisierung gab es aber keine Platz mehr, um z.B. deutsche Umlaute darzustellen. Neue Standards entstanden, welche auf einem 8-Bit-Code basierten und statt 128 Zeichen deren 256, also die doppelte Anzahl abbilden konnten.

Der erweiterte ASCII-Code, EBCDIC und ANSI (8-Bit-Code)

▲ Zurück zum Anfang

Die Entwicklung hin zum **8-Bit-Code** verlief gleich dreigleisig. Microsoft führt für ihr damaliges Betriebssystem **MS-DOS** den **erweiterten ASCII-Code** ein, wobei dieser Name ein Bluff ist, denn bei diesem Code handelt es sich nicht um eine Erweiterung, welche das Institut, welche den ASCII-Code festgelegt hatte eingeführt hatte, vielmehr handelt es sich um einen Alleingang von Microsoft. Parallel dazu entstanden zwei weitere aber herstellerunabhängige Standards, welche auch auf einem 8-Bit-Code beruhen, namentlich **EBCDIC** *(Extended Binary Coded Decimal Interchange Code)*.und **ANSI** (*American National Standards Institute*). Ein 8-Bit-Code ermöglicht es 2^8 Zeichen (das sind **256**) abzubilden.

Im Umfeld der Grossrechner findet man vorwiegend den **EBCDIC-Standard** , wo gegen im Kleinrechnerbereich sich der **ANSI-Standard** durchgesetzt hat. Beide Systeme beruhen auf einem sehr ähnlichen Prinzip, allerdings ist die Zeichenbelegung völlig verschieden. Dies hat zur Folge, dass es eines **Konverters** bedarf, wenn ein Schriftstück welches in ANSI kodiert ist unter EBCDIC betrachtet werden soll oder umgekehrt.

Diese **8-Bit Zeichensätze** sind in zwei gleich grosse Bereiche unterteilt. Der untere Teil (**Position 0 - 127**), welcher sich nochmals unterteilen lässt, nämlich in Steuerzeichen, Buchstaben und Zahlen ist weltweit fix (im ANSI-Code entspricht er übrigens dem alten ASCII-Zeichensatz) und enthält auf den **Positionen 0 - 31** sämtliche gebräuchlichen Steuerzeichen. welche dem Rechner einen Impuls vermitteln können (Zeilenumbrüche, Tab-Befehl etc.). Diese Zeichen sind nicht darstellbar (denn Sie würden ja etwas beim Rechner auslösen) und werden deshalb meist mit einem kleinen Quadrat symbolisiert. Wenn Sie auf Ihrem Rechner trotzdem einen Zeilenumbruch sehen, handelt es sich dabei nicht um dieses Steuerzeichen, sondern um ein Zeichen auf einer anderen Position in der Zeichentabelle, welche dieses Steuerzeichen visualisiert. **Ab Position 32** folgen dann zuerst ein paar wichtige Sonderzeichen (z.B. der Leerschlag), dann Zahlen (aufsteigend), dann Grossbuchstaben (alphabetisch) gefolgt von Kleinbuchstaben (ebenfalls alphabetisch). Der Bereich endet dann mit weiteren weit verbreiteten Sonderzeichen. Diese Reihenfolge ist sehr bewusst so gewählt, ermöglicht diese Folge doch eine effiziente und praxisorientierte Sortierung von Daten. Und was wäre die Datenverarbeitung ohne effiziente Sortiermöglichkeit?!

Mit dem Übergang von MS-DOS nach MS Windows hat übrigens auch Microsoft auf den ANSI-Standard gewechselt. Dieser Wechsel hat zur Konsequenz, dass es beim Öffnen einer DOS-Datei eines Konverters bedarf, damit auch die Zeichen, welche sich in der Zeichentabelle in der oberen Region befinden, richtig angezeigt werden. Wie Sie mit MS Word eine DOS-Datei öffnen können und zwar mit korrekter Darstellung der Umlaute und Sonderzeichen, entnehmen Sie bitte dem Link am Ende dieses Skriptes.

Standard- und Symbolschriften

▲ Zurück zum Anfang

Alles was hier besprochen wurde, wäre und bliebe ein Zahlenfriedhof, gäbe es keine Schriften, welche diese Code in Buchstaben, Zahlen und Sonderzeichen umsetzen könnten, damit auch Menschen, was brauchbares heraus lesen können. Dazu kommen Schriftenarten auch *Fonts* genannt zum Einsatz. Es stehen eine Unzahl Schriftarten zur Verfügung, welche oft wohlklingende Namen wie z.B. *Arial*, *Times New Roman* oder *Bauhaus 93* tragen.

So verschieden diese Schriften auch aussehen mögen, die meisten davon beachten den ANSI-Standard ganz genau. Der Buchstabe "A" mag ja in zwei verschiedenen Schriften ganz individuell daher kommen, es ist und bleibt aber ein "A". Es gibt aber auch (sinnvolle) Ausnahmen . Dazu gehören sicher die diversen Symbolschriften, wie z.B. *Windings* oder *Webdings*. Diese Schriften kümmern sich keinen Deut um den ANSI-Standard, sondern zeigen keck beim betätigen der Taste "A" ein *Smily*-Symbol oder ähnliches an.

Diese Schriftarten sind zwar beliebt, was sie aber vorführen ist eigentlich ein Missbrauch. Der ANSI-Code beruht nun aber mal auf maximal 256 Zeichen (und ähnlichem) und es ist wirklich kein Platz, hier noch alle denkbaren Sonderzeichen sauber zu integrieren. Der ANSI-Code hat also längst seine Möglichkeiten überschritten. Dies gilt um so mehr, als dass mit der Verbreitung des Internets eine Globalisierung statt gefunden hat, welche wohl niemand voraus geahnt hat. Nur ein Teil der im Internet publizierten Dokumente basieren auf einer Sprache, welche lateinische Zeichen bzw. arabische Ziffern verwendet. Wie möchten Sie nun mit einem Zeichensatz, welcher optimistisch gerechnet lediglich 256 Zeichen darstellen kann, zusätzlich zur Lateinischen Schrift noch griechisch, kyrillisch, arabisch hebräisch oder sogar chinesisch schreiben und lesen?

Diese Überlegungen sollen uns zum nächsten Kapitel führen, nämlich dem **Unicode-System**. Sie werden es bereits erraten haben, der Code wird wieder einmal verlängert. Dieses Mal wird aber gleich mit der grossen Kelle angerichtet, aus dem 8-Bit-Code wird nun ein 16-Bit Code. 16 Bits entsprechen dann auch gleich 2 Bytes und damit lassen sich nicht 256-Zeichen anzeigen sondern 256 x 256 Zeichen (bzw. 2^{16} was das selbe ist) und das sind dann über 65'000 Zeichen.

Bevor wir uns aber dem **UCS** (*Universal Character Set*) vom Unicode-Konsortium widmen, fassen wir nochmals zusammen und erweitern unsere Kenntnisse noch etwas.

Die Schichten («*Layers*») im Zusammenhang mit Zeichentabellen

▲ Zurück zum Anfang

Das System baut auf drei **Schichten** («*Layers*») auf, welche bewusst gegenseitig abgegrenzt sind:

- Auf dem untersten «*Layer*», steht ein 1-Byte-Wert (bzw. 2-Byte-Wert im Unicode). Dieser Wert lässt sich hexadezimal mit einem Wert zwischen 00 und FF ausdrücken oder dezimal mit einem Wert zwischen 0 und 255. Für den Unicode gilt hexadezimal 0000 bis FFFF bzw. dezimal 0 bis 65535.
- Der mittlere «*Layer*» ist ein Standard, ausgegeben von einer internationalen Organisation, welches jedem Wert ein Zeichen zuweist. Im PC-Bereich ist der ANSI-Standard sehr verbreitet der Unicode ist aber mit fortschreitender Globalisierung stark im kommen. Die unteren 128 Werte im ANSI-System sind weltweit fix zugeteilt, während die oberen 128 Werte je nach Weltregion verschieden sind. Für diese Werte wurden zusätzliche Standards geschaffen nämlich von der **ISO** in Genf (*International Standard Organisation*). Es handelt sich dabei um die **ISO-8859-x** Zeichnsätze. Der Grund, dass diese ISO-8859-Familien gebildet wurden liegt darin, dass sich zwar sowohl Germanen, Slawen und romanische Völker auf das Lateinische Alphabet berufen, aber andere Bedürfnisse haben, bezüglich der Zeichen im Alphabet, welche sich nach dem Zerfall des Römischen Reiches heraus gebildet haben. So braucht der Deutsche z.B. Umlaute um sich auszudrücken, der Franzose dagegen Buchstaben mit Akzenten. Unter Windows werden diese ISO-8859-Familien übrigens auch **charsets** genannt. Die ISO definiert folgende Familien:
  - iso-8859-1 (**Latin-1**) *Westeuropa*
  - iso-8859-2 (**Latin-2**) *Slawisch mit lateinischer Schrift, Rumänisch, Ungarisch*
  - iso-8859-3 (**Latin-3**) *Türkisch I*
  - iso-8859-4 (**Latin-4**) *Baltische Sprachen*
  - iso-8859-5 *Kyrillisch*
  - iso-8859-6 *Arabisch*
  - iso-8859-7 *Neugriechisch*
  - iso-8859-8 *Hebräisch*
  - iso-8859-9 (**Latin-5**) *Türkisch II*
  - iso-8859-10 (**Latin-6**) *Sami (Grönland und Lappland)*

Sie finden am Ende dieses Skriptes einen Link, welche die Zeichencodetabelle für all diese Familien abbildet. Im Aufbau befinden sich noch zusätzliche Familien wie z.B. Thai, Indisch aber auch Walisisch und Gälisch (was tut man nicht alles für die rund 100'000 Bewohner der Schottischen Highlands).

- Der oberste «*Layer*» bildet die Schriftart, auch Font genannt. Darin ist letztlich festgelegt, wie z.B. ein "A" optisch erscheinen soll. Es gibt eine Unmenge von Schriftarten, welche in einem ganz individuellen «*Look*» daher kommen. In aller Regel, bilden diese substantiell aber das selbe ab. Wichtige Ausnahmen bilden hier Symbolschriften.
- Beim Macintosh-Rechner kommt übrigens nochmals ein anderer Zeichensatz zum Zuge. Hier gilt: die unteren 128 Positionen entsprechen dem ASCII-Code wogegen die oberen 128 Positionen anders vergeben sind, als dies der ANSI-Standard vorsieht. Damit können Dateien, welche auf dem Mac-Rechner erstellt wurden nicht direkt auf einen Rechner übernommen werden, welcher unter Windows oder noch unter MS-DOS betrieben wird. Das selbe gilt natürlich auch im umgekehrten Falle. Es bedarf eines Konverters, soll der Datenaustausch gelingen.

Unicode oder «*Universal Character Set*» (UCS)

▲ Zurück zum Anfang

Hinter dem Unicode steckt einerseits ein globaler Gedanke, nämlich die Möglichkeit neben den alphabetisch aufgebauten Schriftsprachen wie griechisch, lateinisch oder kyrillisch auch Sprachen abbilden zu können, welche nicht auf ein Alphabet mit vielleicht zwei bis drei Duzend Zeichen basieren. So werden z.B. in vielen asiatischen Sprachen mit einem Zeichen eine ganze Silbe dargestellt. Solche Sprachen verfügen dann auch über einen viel grösseren Zeichensatz, als das wir es uns in Europa gewohnt sind.

Im weitern soll das Unicode System, Ordnung schaffen und weltweit zu einem einheitlichen Resultat führen. Es soll auch ermöglicht werden im gleichen Schriftstück zwischen verschiedenen Schriften hin und her zu schalten. Der 16-Bit Code des Unicode Systems bietet Platz für über 65'000 Zeichen. In der Unicode 2.0 Version sind bereits knapp 26'000 davon vom Unicode-Konsortium offiziell vergeben. In der neusten verfügbaren Version 3.0, sind bereits knapp 40'000 Zeichen vergeben. Diese Version enthält dann bereits die Zeichen für so alltägliche Bedürfnisse, wie das Schreiben eines Liebesbriefes mit altgermanischen Runen, oder das Kommunizieren mit den knapp 50'000 verschont gebliebenen Cherokee-Indianer in Oklahoma. Der Rest ist in Reserve für zusätzliche Wünsche oder für den Fall, dass *E.T.* doch mal mit uns kommunizieren sollte. Jedes Zeichen, welches neu aufgenommen werden soll muss beantragt und von der Organisation abgesegnet werden.

Offene Probleme im Zusammenhang mit dem Unicode

▲ Zurück zum Anfang

Es sind vor allem zwei Probleme, welche den Benutzer von Unicode-Zeichen beschäftigen. Das eine ist, dass es nicht möglich ist, z.B. unter Windows ein Unicode-Zeichen über die Tastatur einzugeben. Man bräuchte dazu Hilfsmittel, welche auf dem Bildschirm eine virtuelle Tastatur anzeigen und die Zeichen über einen Mausklick erreichbar machen. Die Tools, welche ich dafür kenne, kommen von Microsoft und sind kostenpflichtig (z.B. das *Office2000 Multi Language Pack* (MLP).

Diese Seite enthält einen Link zu einem Skript, welches aufzeigt, wie man mit Hilfe einer VBA-Routine einen Unicode in ein MS Word-Dokument einbringen kann.

Das zweite Problem, welches anzusprechen ist, betrifft die Schriftarten (*Fonts*), welche all diese Zeichen anzeigen können. Diese Schriften sind rar. Microsoft stellt aber etwa acht Schriften zur Verfügung, welche die in der westlichen Welt gebräuchlichen Zeichen darstellen können. Die Namen dieser Schriften lauten gleich wie die bekannten Namen, also z.B. *Arial*. Allerdings bräuchten Sie dazu auch die richtige Version, wenn Sie mit Unicodes arbeiten möchten. Am Ende dieses Skriptes befindet sich ein Link, welcher es Ihnen erlaubt diese Schriften herunter zu laden. Haben Sie die Schriften herunter geladen, können Sie diese mittels Maus-Doppelklick auf einfachste Weise installieren. Nach einem Neustart des Rechners, stehen dann für diese populären Schriften die Unicode-Zeichen zu Ihrer Verfügung. Diese Schriftdateien legen dabei übrigens markant an Grösse zu.

Ich kenne genau eine Schrift, welche wirklich alle vergebenen Unicode-Zeichen darstellen kann. Leider ist diese Schrift noch auf dem Unicode **Version 2.0** Standard. Das mit dem Liebesbrief an den germanischen Jüngling muss also noch auf sich warten lassen. Allerdings kann diese Schriftart die Zeichen vieler zentral- und ostasiatische Schriftsprachen darstellen. Diese Schrift nennt sich **Arial Unicode MS**. Die Schrift ist leider nicht frei verfügbar, sondern muss z.B. im Zusammenhang mit dem *MS Office2000* erworben werden. Diese Schriftdatei ist übrigens ganze **23** MB schwer. Sie finden diese Schrift z.B. auf der MS Office2000 **CD 2** und zwar unter dem Namen **Arialuni.ttf** in komprimierter Form im **OFFCD2_1.CAB**.

Sie können diese Schriftart allerdings auch über den Office-Setup installieren, standardmässig wird die Schrift allerdings nicht installiert. Wie schon erwähnt ist dies ausschliesslich unter **Office2000** möglich. Gehen Sie wie folgt vor:

Unicode Schriftart installieren

▲ Zurück zum Anfang

1. Schmeissen Sie Ihre Office2000-CD in das Laufwerk.
2. Wählen Sie ***Start - Einstellungen - Systemsteuerung - Software***.
3. Wählen Sie in der Liste das Produkt **Microsoft Office2000** aus.
4. Klicken Sie auf die Schaltfläche **Hinzufügen/Entfernen**.
5. Klicken Sie den Knopf **Features hinzufügen/Entfernen** an.
6. Erweitern Sie die Auswahl unter **Office Tools**.
7. Erweitern Sie nun die Auswahl unter **Internationale Unterstützung**.
8. Klicken Sie auf den Eintrag **Universale Schriftart** und wählen Sie im Kontextmenü **Vom Arbeitsplatz starten**.
9. Klicken Sie auf die Sachtfläche **Jetzt aktualisieren** und schliessen Sie den Setup ordnungsgemäss ab.

Wie erfahre ich den Code eines Unicode-Zeichens?

▲ Zurück zum Anfang

Am Schluss dieses Skriptes finden Sie einen Link, welcher Sie zu einer **PDF-Datei** führt, welche die meisten Unicode-Zeichen, tabellarisch und nach Kategorien geordnet samt dem dazugehörigen Hexadezimal- und Dezimalcode auflistet. Da diese Datei ein PostScript ähnliches Format hat. sollten Sie die Zeichen erkennen, unabhängig davon, ob Sie eine Unicode-fähige Schrift installiert haben oder nicht.

Sie finden zusätzlich, einen Link zu einem Skript, welches eine VBA-Routine beinhaltet, die es Ihnen erlaubt, in Word eine Tabelle für eine fast beliebige Uncode-Zeichenkategorie zu erstellen. Das Aussehen der Tabelle entspricht dem, was Sie im PDF-Dokument sehen können. Ich möchte nochmals erwähnen, dass Sie dazu die neuste Version .z.B. der Schrift *Arial* benötigen. Diese können Sie über den unten aufgeführten Link bei Microsoft herunterladen. Um auch die *exotischen* Schriften anzuzeigen, gibt es leider keine kostengünstige Gesamtlösung. Viele Universitäten und Sprachinstitute bieten im Internet aber Schriften an, welche es erlauben, einzelne Schriftsprachen anzuzeigen. Beachten Sie bitte, die jeweiligen *Copyright* Bestimmungen. Nicht jeder Schrift ist Allgemeingut.

Zeichen, welche nicht belegt sind oder die zur Verfügung stehende Schriftart nicht anzeigen kann, erscheinen übrigens als kleine Quadrate.

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

- ► Wie kann eine Tabelle sämtlicher ANSI (ASCII) Codes umgesetzt in einer bestimmten Schriftart oder Symbolschrift erstellen?
- ► Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen?
- ► Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?
- ► Die Umlaute in einem DOS-Text werden beim Öffnen unter Word falsch dargestellt

Microsoft

- ► Unicode taugliche TrueType Schriften

Stefan Münz

- ► Zeichensätze (iso-8859-Familie und andere)

Das Unicode Konsortium

- ► Unicode Home Page

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/ZCode.htm

René Probst, Mai 2000

Änderungen

- 25. Januar 2001 - Unversale Schriftart über Office-Setup installieren

Wie lautet der ANSI-Wert zu den in Word gebräuchlichen Sonderzeichen, wie geschütztes Leerzeichen geschützter Trennstrich, etc.

Wie kann ich den ANSI (ASCII) Code eines (Sonder-)Zeichens in einem Word-Dokument ermitteln?

Wie kann eine Tabelle sämtlicher ANSI (ASCII) Codes umgesetzt in einer bestimmten Schriftart oder Symbolschrift erstellen?

◄ Zurück zum Hauptindex

Aufgabenstellung

> 1. Ich möchte mittels VBA einen geschützten Leerschlag in mein Dokument einfügen. Dafür brauche ich wohl zu erst den ANSI-Code (*American National Standards Institute*) für dieses Zeichen. Wie kann ich diesen Wert ermitteln?
>
> 2. Ich möchte für eine bestimmte Schrift sämtliche Zeichen drucken und für diese Zeichen den dazu gehörigen ANSI-Wert auflisten.

Diese Seite

- ▼ Einleitung
- ▼ ANSI-Code für die meist gebräuchlichen Word-internen Sonderzeichen
- ▼ ANSI-Wert einer markierten Zeichenfolge anzeigen
- ▼ Tabelle mit allen potentiell zur Verfügung stehenden Zeichen inklusive dem dazugehörigen ANSI-Wert generieren
- ▼ ANSI-Zeichen über die Tastatur eingeben
- ▼ Umrechnen einer hexadezimalen Zahl in das Dezimalsystem
- ▼ Der Tastaturtreiber und seine Rolle
- ▼ Links zu diesem Thema

Einleitung

▲ Zurück zum Anfang

Im folgenden finden Sie zuerst eine Tabelle, welche den ANSI-Code für die meist gebräuchlichen Word-internen Sonderzeichen auflistet.

Anschliessend werden mehrere VBA-Lösungen präsentiert. Die erste Routine zeigt den ANSI-Wert eines im Dokument markierten Zeichens (bzw. einer Folge von markierten Zeichen) an; die zweite Routine erstellt in einem Dokument eine Tabelle mit allen Zeichen. Aus dieser Tabelle können Sie den ANSI-Wert für jedes Zeichen auf einfache Weise herauslesen.

Sie erfahren weiter in diesem Skript, wie sich eine hexadezimale Zahl in eine Dezimalzahl umrechnen lässt. Das Dokument enthält auch einen Link, zu einem anderen Beitrag, welcher interessante und spannende Informationen rund um die Zeichensatztabellensuppe bereit hält.

ANSI-Code für die meist gebräuchlichen Word-internen Sonderzeichen

▲ Zurück zum Anfang

Sonderzeichen		Code	Hex	Shortcut
Absatz als Zeichen	¶	182	B6	Enter
Auslassungspunkte	…	133	85	Alt+Strg+. (Punkt)
Bedingter Trennstrich	¬	31	1F	Strg+- (Bindestrich)
Copyright	©	169	A9	Alt+Strg+C
Einfaches öffnendes Anführungszeichen	‘	145	82	AutoKorrektur
Einfaches schliessendes Anführungszeichen	’	146	92	AutoKorrektur
Eingetragene Marke	®	174	AE	Alt+Strg+R
Gedankenstrich	–	150	96	Strg+NUM+ -
Geschützter Trennstrich	–	30	1E	Strg+-
Geschütztes Leerzeichen	°	160	A0	Strg+Umschalt+Leertaste
Geviertstrich	—	151	97	Alt+Strg+Num -
Markenzeichen	TM	153	99	Alt+Strg+T
Öffnendes Anführungszeichen	“	132	84	AutoKorrektur
Paragraphenzeichen	§	167	A7	§
Schliessendes Anführungszeichen	”	147	93	AutoKorrektur

ANSI-Wert einer markierten Zeichenfolge anzeigen

▲ Zurück zum Anfang

Die nachfolgende Routine zeigt den ANSI-Wert eines im Dokument markierten Textes an. Für die Anzeige wird primär der Office-Assistent herangezogen. Sie können den Wert allerdings auch in tabellarischer Form betrachten, wenn Sie die Schaltfläche **Weiter** betätigen.

Für das letztere gelangt eine *UserForm* zur Anwendung. Alle Schaltflächen dieser UserForm erstellt der Code aber dynamisch. Sie müssen also keine Schaltflächen zeichnen, sondern können von einer leeren UserForm ausgehen. Um diese Lösung in Ihrer Umgebung einzubringen, gehen Sie bitte wie folgt vor:

Implementierung der Lösung

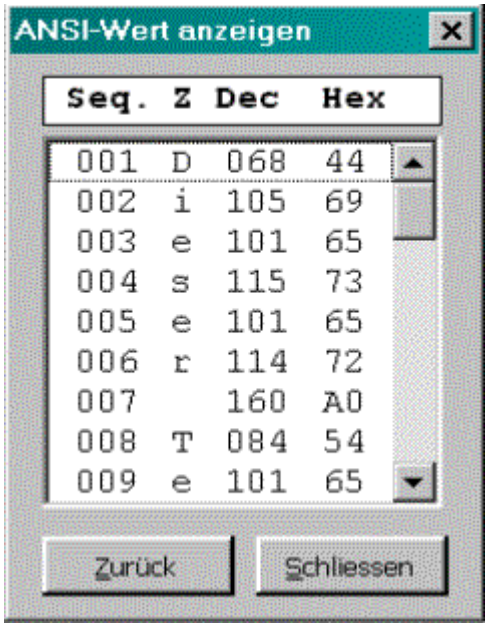
1. Öffnen Sie die Standardvorlage mit dem Namen **Normal.dot** (Rechtsklick im Windows-Explorer - Kontextmenü - Öffnen).
2. Wechseln Sie mit der Tastenkombination **[Strg + F11]** in die VBA-Umgebung.
3. Wählen Sie im VBA-Editor im Hauptmenü den Dialog ***Einfügen - Modul*** und kopieren Sie den kurzen Code, welcher unten abgebildet ist, in das leere Code-Fenster.
4. Wählen Sie nun im Hauptmenü den Dialog ***Einfügen - UserForm***.
5. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt **Code anzeigen**.
6. Löschen Sie nun die zwei Code-Zeilen, welche von Word automatisch generiert wurden und kopieren Sie den grösseren Code, welcher unten abgebildet ist in das nun leere Fenster.
7. Lassen Sie sich nun mit der Taste **[F4]** den Eigenschaftsdialog der UserForm anzeigen.
8. Ändern Sie nun die Eigenschaft **Name** von *UserFormx* nach **frmANSIWert**.
9. Speichern Sie nun die Vorlage und verlassen Sie Word.
10. Starten Sie Word wieder, markieren Sie etwas Text im Dokument und testen Sie die Sache.

VBA-Code zum Aufruf der UserForm

```
Sub ANSIWert()
  Set oAssistant = Assistant
  If oAssistant Is Nothing Then
    MsgBox "Der Office-Assistent ist nicht oder nicht korrekt installiert.", _
       vbCritical
    End
  End If
  Set oAssistant = Nothing
  frmANSIWert.ANSIWertProlog
End Sub
```

(Ruft die UserForm mit dem Namen frmANSIWert auf.)

VBA-Code für die UserForm




```
Private Const asTitel = "ANSI-Wert anzeigen"
Private Const asText = _
   "Diese Routine zeigt den ANSI-Wert für eine
markierte Zeichenfolge an."
Private Const MaxLen = 999
Private Titel As String
Private Text As String
Private strSel As String
Private strNums As String
Private lstr As Long
Private asAssistant As Boolean
Private lblHeader As MSForms.Label
Private List As MSForms.ListBox
Private WithEvents CmdBack As MSForms.CommandButton
Private WithEvents CmdClose As MSForms.CommandButton

Private Sub ANSIWertAnzeigenTabelle()
  With Me
    .Width = 135
    .Height = 186
    .Caption = asTitel
  End With
  Set lblHeader = Controls.Add("Forms.Label.1", "", True)
  With lblHeader
    .Top = 6
    .Left = 9
    .Width = 112
    .Height = 15
    .BackColor = &HFFFFFF
    .BorderStyle = fmBorderStyleSingle
    .Font.Name = "Courier New"
    .Font.Bold = True
    .Font.Size = 10
  End With
  Set List = Controls.Add("Forms.ListBox.1", "", True)
  With List
    .Top = 24
    .Left = 9
    .Width = 112
    .Height = 120
    .Font.Name = "Courier New"
    .Font.Size = 10
    .ColumnCount = 4
    .TextAlign = fmTextAlignLeft
    .ColumnWidths = "27;15;28;15"
  End With
  Set CmdBack = Controls.Add("Forms.Commandbutton.1", "CmdBack", True)
  With CmdBack
    .Top = 144
    .Left = 9
    .Width = 54
    .Height = 18
    .Caption = "Zurück"
    .Accelerator = "Z"
  End With
```

```
  Set CmdClose = Controls.Add("Forms.Commandbutton.1", "Cmdclose", True)
  With CmdClose
    .Top = 144
    .Left = 69
    .Width = 54
    .Height = 18
    .Caption = "Schliessen"
    .Accelerator = "S"
  End With
End Sub

Sub ANSIWertProlog()
  asAssistant = Assistant.Visible
  Assistant.Visible = True
  Assistant.Animation = msoAnimationGreeting
  If Selection.Type <> 2 Then
    ANSIWertNichtsMarkiert
  Else
    ANSIWertErmitteln
  End If
  Assistant.Visible = asAssistant
End Sub

Private Sub ANSIWertErmitteln()
  Titel = asTitel
  strSel = Selection.Text
  lstr = Len(strSel)
  strNums = ""
  Select Case lstr
    Case 1
      Text = "Für das markierte Zeichen wurde folgender ANSI-Wert ermittelt:"
    Case Is > MaxLen
      strSel = Left(strSel, MaxLen)
      Titel = asTitel & vbCrLf & vbCrLf & "Markierter Text zu lang! _
         (" & lstr & " Zeichen)"
      lstr = MaxLen
      Text = "Für die ersten " & MaxLen & " Zeichen des markierten Text _
         wurde folgender ANSI-Wert ermittelt:"
    Case Else
      Text = "Für den markierten Text (" & lstr & " Zeichen) _
         wurde folgender ANSI-Wert ermittelt:"
  End Select
  i = 1
  For i = 1 To lstr
    strNums = strNums & Asc(Mid(strSel, i)) & " "
  Next i
  ANSIWertAnzeigen
End Sub

Private Sub ANSIWertAnzeigen()
  Dim lstNums As String
  Dim Blase As Balloon
  Set Blase = Assistant.NewBalloon
  With Blase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeBullets
    .Icon = msoIconTip
    .Heading = asTitel
```

```
    .Text = Text & vbCrLf & vbCrLf & strNums
    .Button = msoButtonSetNextClose
  End With
  Antwort = Blase.Show
  Select Case Antwort
    Case -6
      ANSIWertAnzeigenTabelle
      lblHeader.Caption = " Seq. Z Dec  Hex"
      ReDim c(lstr - 1, 3)
      For i = 0 To lstr - 1
        c(i, 0) = Format(i + 1, "000")
        c(i, 1) = Mid(strSel, i + 1, 1)
        c(i, 2) = Format(Asc(c(i, 1)), "000")
        If Val(c(i, 2)) < 32 Then c(i, 1) = Chr(127)
        c(i, 3) = Hex(c(i, 2))
        If Len(c(i, 3)) = 1 Then c(i, 3) = "0" & c(i, 3)
      Next i
      List.List() = c()
      Me.Show
  End Select
End Sub

Private Sub ANSIWertNichtsMarkiert()
  Assistant.Animation = msoAnimationGetAttentionMinor
  Dim Blase As Balloon
  Set Blase = Assistant.NewBalloon
  With Blase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeNumbers
    .Icon = msoIconAlert
    .Heading = asTitel & vbCrLf & vbCrLf & "Kein Text markiert !"
    .Text = "Sie haben leider kein Text markiert  - so wirds gemacht:"
    .Labels(1).Text = "Brechen sie diese Prozedur ab"
    .Labels(2).Text = "Markieren Sie ein Zeichen oder den gewünschten Text."
    .Labels(3).Text = "Starten sie diese Prozedur erneut."
    .Button = msoButtonSetCancel
  End With
  Blase.Show
End Sub

Private Sub CmdBack_Click()
  Me.Hide
  ANSIWertAnzeigen
End Sub

Private Sub CmdClose_Click()
  Unload Me
End Sub
```

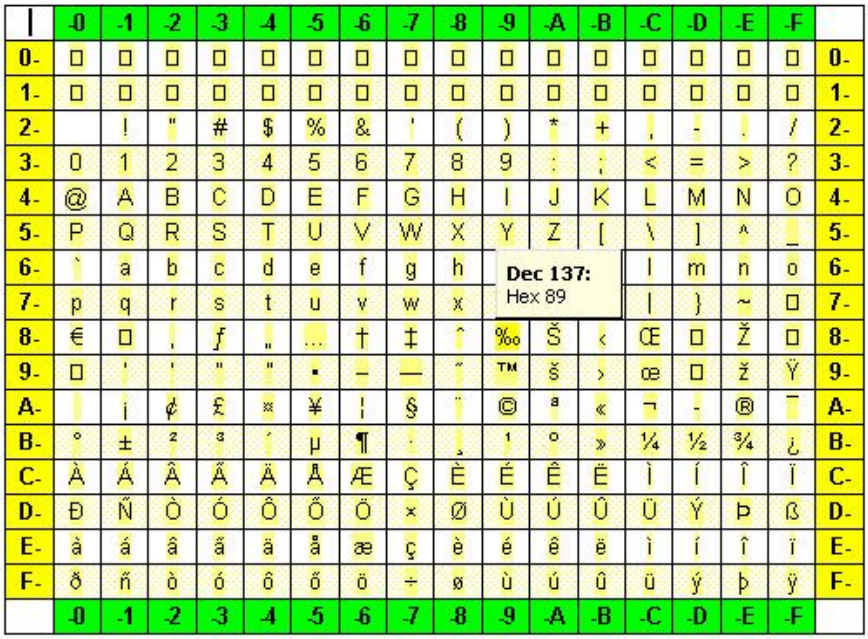
(Gibt den ANSI-Codes eines markierten Zeichens oder einer markierten Zeichenfolge zurück. Auf Wunsch auch in tabellarischer Darstellung.)

Tabelle mit allen potentiell zur Verfügung stehenden Zeichen inklusive dem dazugehörigen ANSI-Wert generieren

▲ Zurück zum Anfang

Folgende VBA-Routine generiert eine Tabelle mit all jenen Zeichen, welche potentiell für eine Schrift zur Verfügung stehen. Aus dieser Tabelle können Sie für jedes Zeichen den ANSI-Code herauslesen. Primär ermitteln Sie damit den hexadezimalen Wert des ANSI-Codes. Wie Sie eine hexadezimale Zahl in das Dezimalsystem umrechnen können, erfahren Sie übrigens etwas weiter unten.

Wenn Sie mit der Maus aber über eines der Zeichen fahren, erscheint sowohl der dezimale, wie auch der hexadezimale Wert des Zeichens. Um die Routine anzuwenden, gehen Sie wie folgt vor:



	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	
0-	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	0-
1-	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	1-
2-		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	2-
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	3-
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	4-
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	5-
6-	`	a	b	c	d	e	f	g	h				l	m	n	o	6-
7-	p	q	r	s	t	u	v	w	x				|	}	~	□	7-
8-	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	Ž	□	8-
9-	□	'	'	"	"	•	–	—	˜	™	š	›	œ	□	ž	Ÿ	9-
A-		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	A-
B-	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	B-
C-	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	C-
D-	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	D-
E-	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	E-
F-	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	F-
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	

Implementierung

1. Kopieren Sie den nachfolgenden Code in ein neues Modul der Standardvorlage mit dem Namen **Normal.dot**.
2. Wählen Sie im Dokument über das Kombinationsfeld **Schriftart** auf der Symbolleiste mit dem Namen *Format* die Schrift Ihrer Wahl.
3. Lassen Sie das Makro mit dem Namen **ZeichenTabelle** ausführen.
4. Das Programm wird Sie nun fragen, ob *Quick-Infos* für die Zeichen erstellt werden sollen. Wenn Sie dies Frage bejahen, fügt dar Code für jedes Zeichen in der Tabelle ein Kommentar ein. Dieser Kommentar enthält sowohl den dezimalen, wie auch den hexadezimalen ANSI-Code des Zeichens. Wenn Sie mit der Maus über eines der Zeichen in der Tabelle fahren, erscheint dann automatisch als *Quick-Info*, diese nützliche Information am Bildschirm. Das Generieren dieser Kommentare dauert etwas. Grundsätzlich müssen Sie diese Tabelle aber nur einmal erstellen. Sie können die generierte Tabelle dann auf Ihrem Rechner abspeichern. Sollten Sie diese Frage verneinen, werden diese Kommentare nicht eingefügt. Sie können in diesem Fall ausschliesslich an Hand der Position des Zeichens in der Tabelle ablesen, welchen hexadezimalen ANSI-Code das Zeichen trägt. Ohne diese Kommentare, ist das Makro schnell.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Titel = "Zeichentabelle erstellen"

Sub ZeichenTabelle()
  Text = "Soll die Tabelle mit Quick-Infos erstellt werden?" & vbCrLf & vbCrLf
  Text = Text & "Wenn Sie diese Frage mit Ja beantworten, kann es etwas dauern."
  Antwort = MsgBox(Text, vbYesNoCancel, Titel)
  If Antwort = 2 Then End
  System.Cursor = wdCursorWait
  Dim nDoc As Document
```

```
Dim Zeile As Row, Spalte As Column, Zelle As Cell
Dim Objekt As Object
Dim x As Integer
Dim y As Integer
With Selection
  sName = .Font.Name
  sFett = .Font.Bold
  sKursiv = .Font.Italic
  sUnterstrichen = .Font.Underline
End With
Set nDoc = Documents.Add
Application.ScreenUpdating = False
For i = 0 To 15
  Kopf = Kopf & Chr(1) & "-" & Hex(i)
Next i
With Selection
  .TypeText Text:=Kopf & vbCrLf
  While y <= 15
    Rahmen = Hex(y) & "-" & Chr(1)
    Selection.TypeText Text:=Rahmen
    Data = ""
    x = 0
    While x <= 15
       Code = y * 16 + x
       If Code < 32 Then Code = 127
       Data = Data & Chr(Code) & Chr(1)
       x = x + 1
    Wend
    .TypeText Text:=Data
    Rahmen = Hex(y) & "-"
    .TypeText Text:=Rahmen & vbCrLf
    y = y + 1
  Wend
  .TypeText Text:=Kopf & vbCrLf
  .WholeStory
  .Font.Size = 12
  .Font.Name = sName
  .Font.Bold = sFett
  .Font.Italic = sKursiv
  .Font.Underline = sUnterstrichen
  .ConvertToTable Separator:=Chr(1), Format:=wdTableFormatNone
End With
With nDoc.Tables(1)
  .Range.ParagraphFormat.Alignment = wdAlignParagraphCenter
  .Range.Cells.Height = "18"
  .Range.Cells.VerticalAlignment = wdCellAlignVerticalCenter
  i = 3
  While i <= 18
    rest = i Mod 2
    If rest = 1 Then
      .Rows(i).Cells.Shading.ForegroundPatternColorIndex = wdYellow
      .Rows(i).Cells.Shading.Texture = wdTexture20Percent
    End If
```

```
      i = i + 1
    Wend
    ZellFarbe .Rows(1), wdBrightGreen, wdBrightGreen, wdTexture40Percent
    ZellFarbe .Rows(18), wdBrightGreen, wdBrightGreen, wdTexture40Percent
    ZellFarbe .Columns(1), wdYellow, wdYellow, wdTexture40Percent
    ZellFarbe .Columns(18), wdYellow, wdYellow, wdTexture40Percent
    ZellFarbe .Cell(1, 1), wdWhite, wdWhite, wdTextureSolid
    ZellFarbe .Cell(1, 18), wdWhite, wdWhite, wdTextureSolid
    ZellFarbe .Cell(18, 1), wdWhite, wdWhite, wdTextureSolid
    ZellFarbe .Cell(18, 18), wdWhite, wdWhite, wdTextureSolid
    .Rows(1).Select
    ZeichenTabelleRahmenschrift
    .Rows(18).Select
    ZeichenTabelleRahmenschrift
    .Columns(1).Select
    ZeichenTabelleRahmenschrift
    .Columns(18).Select
    ZeichenTabelleRahmenschrift
  End With
  If Antwort = 6 Then ZeichenTabelleKommentareEinfügen
  Selection.HomeKey unit:=wdStory
  ActiveWindow.WindowState = wdWindowStateMaximize
  ActiveWindow.ActivePane.View.ShowAll = False
  Application.ScreenUpdating = True
  System.Cursor = wdCursorNormal
  Set oAssistant = Assistant
  If Not oAssistant Is Nothing Then
     Assistant.Animation = msoAnimationCharacterSuccessMajor
  End If
  Set oAssistant = Nothing
End Sub

Private Sub ZeichenTabelleKommentareEinfügen()
  Dim Zelle As Cell, x As Comment
  Dim i As Integer, H As String
  i = -1
  For Each Zelle In ActiveDocument.Tables(1).Range.Cells
    If Zelle.RowIndex > 1 And Zelle.RowIndex < 18 And _
       Zelle.ColumnIndex > 1 And Zelle.ColumnIndex < 18 Then
      i = i + 1
      If i < 16 Then H = "0" & Hex(1) Else H = Hex(i)
      Set x = ActiveDocument.Comments.Add(Range:=Zelle.Range, _
         Text:="Hex " & Hex(i))
      x.Author = "Dec " & i
    End If
    StatusBar = i + 1 & " Kommentare eingefügt... Bitte warten..."
    DoEvents
  Next
End Sub

Private Function ZellFarbe(x As Object, VG As Long, HG As Long, i As Long)
  If TypeOf x Is Cell Then
    x.Shading.ForegroundPatternColorIndex = VG
```

```
    x.Shading.BackgroundPatternColorIndex = HG
    x.Shading.Texture = i
  Else
    x.Cells.Shading.ForegroundPatternColorIndex = VG
    x.Cells.Shading.BackgroundPatternColorIndex = HG
    x.Cells.Shading.Texture = i
  End If
End Function

Private Function ZeichenTabelleRahmenschrift()
  With Selection.Font
    .Name = "Arial"
    .Bold = True
    .Italic = False
    .Underline = False
  End With
End Function
```

(Erstellt eine Tabelle mit allen potentiell verfügbaren Zeichen einer Schrift. Aus dieser Tabelle können Sie dann den ANSI-Code für ein bestimmtes Zeichen herauslesen. Auf Wunsch erstellt das Programm auch *Quick-Infos*, welche den dezimalen- und hexadezimalen Wert eines Zeichens anzeigen, wenn die Maus auf einem der Zeichen in der Tabelle verharrt.)

ANSI-Zeichen über die Tastatur eingeben

▲ Zurück zum Anfang

Wenn Sie den ANSI-Code des Zeichen wissen, können Sie das Zeichen wie folgt über Ihre Tastatur eingeben:

1. Schalten Sie auf Ihrer Tastatur die Taste, welche mit **[NumLock]** bezeichnet ist ein. Sie sollten auf der Tastatur an Hand einer LED-Anzeige sehen, dass diese Taste nun aktiviert ist.
2. Drücken Sie die linke **[Alt]**-Taste auf Ihrer Tastatur und halten Sie diese gedrückt.
3. Tippen Sie nun den dezimalen Code des Zeichens ein. Die Zahl, welche Sie aus der Tabelle herauslesen, ist entweder zwei- oder dreistellig. Sie müssen allerdings eine **vierstellige** Zahl eingeben. Füllen Sie die Zahl also links mit führend Nullen auf. D. h. für 58 geben Sie 0058 oder für 223 geben Sie 0223 ein. Und noch etwas ist ganz wichtig. Verwenden Sie bei der Eingabe der vierstelligen Zahl nicht die Ziffern in der obern Reihe Ihrer Tastatur, sondern brauchen Sie dazu ausschliesslich den **Ziffernblock**, welcher sich in der Regel im rechten Bereich der Tastatur befindet.
4. Lassen Sie nun die Alt-Taste wieder los und das gewünschte Zeichen wird erscheinen.
5. Sie können nun die NumLock-Taste wieder ausschalten.

Umrechnen einer hexadezimalen Zahl in das Dezimalsystem

▲ Zurück zum Anfang

Als Kopfrechnung

Zuerst sollten Sie wissen, dass das Hexadezimalsystem nicht mit 10 Ziffern auskommt sondern deren 16 benötigt. Man ging hier pragmatisch vor und hat keine neuen Zeichen erfunden, sondern missbraucht dazu einfach die ersten Buchstaben des Alphabetes. Die Ziffern 0 - 9 sind identisch mit dem Dezimalsystem, anschliessend geht es wie folgt weiter:

- 10 = A
- 11 = B
- 12 = C
- 13 = D
- 14 = E
- 15 = F

Der hexadezimale Code für ein Zeichen ist immer zweistellig. Nun gilt es zu beachten, dass Sie die erste Stelle nicht wie im Dezimalsystem mit 10 multiplizieren sollten, sondern mit 16. Hier einige Beispiele, um dies einzuüben;

Hexadezimal	Berechnung	Ergebnis (Dezimalwert)
07	(0 x 16) + 7	7
0C	(0 x 16) + 12	12
10	(1 x 16) + 0	16
1D	(1 x 16) + 13	29
47	(4 x 16) + 7	71
6F	(6 x 16) + 15	111
B8	(11 x 16) + 8	184
FF	(15 x 16) + 15	255

Mittels Windows-Rechner

Sie können auch den in Windows integrierten Rechner benutzen um zum Dezimalwert einer Hexadezimalzahl zu gelangen. Gehen Sie wie folgt vor:

1. Klicken Sie ***Start - Programme - Zubehör - Rechner***.
2. Wenn der Rechner aufgestartet ist, schalten Sie im Menüpunkt **Ansicht** die Ansicht **Wissenschaftlich** ein.
3. Aktivieren Sie im oberen Teil des Dialoges das Optionsfeld mit der Bezeichnung **Hex**.
4. Tippen sie nun den hexadezimalen Code entweder über Ihre Tastatur oder mittels der virtuellen Tastatur auf dem Dialog ein.
5. Aktivieren Sie zum Schluss im oberen Teil des Dialoges das Optionsfeld mit der Bezeichnung **Dec**.

Der Tastaturtreiber und seine Rolle

▲ Zurück zum Anfang

Sollten Sie an Ihrem Bildschirm nicht jenes Zeichen sehen, welches Sie vermeintlich über Ihre Tastatur eintippen, dann kann das daran liegen, dass Sie den falschen Tastaturtreiber, also einen Treiber, welche für eine andere Sprache bzw. eine "fremdländische" Tastatur bestimmt ist, installiert oder aktiviert haben. In diesem Falle sollten Sie unter Ihrem Betriebssystem (oft Windows) den richtigen Treiber bereitstellen. Falls unter Windows zwar alles Ok ist, nicht aber wenn Sie in der MS-DOS Oberfläche arbeiten, dann haben Sie in den Startdateien mit den Namen *config.sys* und/oder *autoexec.bat* den falschen Tastaturtreiber eingebunden. Solange Sie aber eh nie unter DOS arbeiten, hat das letztere keine Auswirkung für Sie direkt. Sie werden höchsten jemanden schimpfen hören, welcher auf DOS-Ebene Ihren Rechner flicken sollte, nachdem Windows wieder einmal bei einem *Crash* ein Chaos zurück gelassen hat.

Wie Sie einen Tastaturtreiber unter Windows installieren können, erfahren Sie in einem der unten eingeblendeten Links (Sprachunterstützung).

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Wissenswertes über die Zeichensatztabellensuppe
► Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen?
► Wie kann ich unter Windows die Sprachunterstützung für baltisch, kyrillisch, griechisch oder türkisch installieren?
► Die Umlaute in einem DOS-Text werden beim Öffnen unter Word falsch dargestellt.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/ANSI.htm

René Probst, Juni 2000

Änderungen

- --- nil ---

Wo kann ich den Unicode eines Zeichens finden, bzw. eine Tabelle für eine Unicode-Familie erstellen?

◄ Zurück zum Hauptindex

Aufgabenstellung

> 1. Wo finde ich die Unicode-Zeichen-Tabellen?
>
> 2. Ich möchte eine Tabelle mit allen kyrillischen Zeichen, welche der UCS (*Universal Character Set*) definiert mit dem dazugehörigen numerischen Wert erstellen.

Einleitung

Sie erfahren in diesem Skript, wie Sie direkt aus **Word2000** ein Unicode-Zeichen in das Dokument einbringen können. Dies ist sowohl über den Dialog ***Einfügen - Symbol... - Symbole*** wie auch mit Hilfe von Word-VBA möglich. **Word 97** unterstützt Unicode leider noch nicht.

Sie benötigen dazu die neuste Version .z.B. der Schrift *Arial*. Diese Schriftdateien können Sie über den unten aufgeführten Link bei Microsoft herunterladen. Die herunter geladenen Schriftarten können die in der westlichen Welt gebräuchlichen oder populären Sprachfamilien darstellen. Die einzige Schrift, welche alle der ca. 28'000 vergebenen Zeichen des **Universal Character Sets 2.0** darstellen kann heisst **Arial Unicode MS** und ist leider nicht frei erhältlich, sondern muss z.B. mit dem MS **Office2000** Programmpaket erworben werden. Beachten Sie weiter, dass Sie diese Schrift allenfalls nachinstallieren müssen. Die Schrift trägt den Dateinamen **ARIALUNI.TTF**. Wie Sie eine Schrift ordnungsgemäss installieren können, erfahren Sie gleich anschliessend. Für Office2000 Anwender erübrigt sich also der Download von Unicode-fähigen Schriften. Microsoft stellt z.Z. leider keine Schrift zur Verfügung, welche Zeichen, die mit UCS 3.0 neu hinzugekommen sind, unterstützt.

Am Schluss dieses Skriptes finden Sie einen Link, der Sie zu einer Datei im PDF-Format führt, welche die Unicode-Zeichen, für fast alle Sprachefamilien, deren Schrift auf einer Art Alphabet beruht, also die Zeichen nicht ganze Silben ausdrücken, tabellarisch und nach Schriftfamilien geordnet samt dem dazugehörigen Hexadezimal- und Dezimalcode, auflistet.

Der Link, welcher zum Kapitel mit dem Name ***Wissenswertes über die Zeichensatztabellensuppe*** führt enthält weitere Informationen zu diesem Thema. Insbesondere, wenn Sie mit dem Gedanken spielen, auch asiatischen Silbensprachen darzustellen, sollten Sie sich dieses Skript zuvor anschauen.

Unicode-Unterstützung unter Office2000 installieren

Bei einer standardmässigen Installation von Office2000, wird die Unicode-Unterstützung nicht installiert. Sie können dies aber jederzeit nachholen. Gehen Sie dabei wie folgt vor:

1. Schmeissen Sie Ihre Office2000-CD ins Laufwerk. Sollten Sie mehrere Scheiben besitzen - es ist die erste der Reihe.
2. Klicken Sie ***Start - Einstellungen - Systemsteuerung - Software***.
3. Suchen Sie in der erscheinenden Liste nach **Microsoft Office2000...** und markieren Sie diesen Einträg.
4. Klicken Sie auf **Hinzufügen/Entfernen**.
5. Im Wartungsmodus angekommen klicken Sie auf **Features Hinzufügen/Entfernen**.
6. Erweitern Sie sie die Auswahl indem Sie zuerst auf **Office Tools** und dann auf **Internationale Unterstützung** klicken.
7. Klicken Sie nun auf **Universalschriftart** und wählen Sie im Kontextmenü **Vom Arbeitsplatz starten**.
8. Betätigen Sie abschliessend den Schaltknopf J**etzt aktualisieren**.

Schriften ordnungsgemäss installieren

Sollten Sie mit Office2000 arbeiten und die Unicode-Unterstützung wie im vorhergehenden Kapitel beschrieben installiert haben, erübrigt sich die Installation von zusätzlichen Schriften, welche Unicode-fähig sind. Falls Sie trotzdem den Wunsch haben, eine Schrift zu installieren, dann gehen Sie wie folgt vor.

1. Wählen Sie ***Start - Einstellungen - Systemsteuerung***.
2. Führen Sie einen Doppelklick auf das Ikon **Schriftarten** aus.
3. Wählen Sie im nun erscheinenden Fenster den Befehl ***Datei - Neue Schriftart installieren...***.
4. Legen Sie den Datenträger mit der Schriftdatei ein, wechseln Sie in das richtige Verzeichnis, markieren Sie die Schriftendatei und betätigen Sie die Schaltfläche ***Ok***.

! Für viele Anwendungen genügt es, die Schrift-Datei in den Schriftenordner zu kopieren. Word und einige andere Anwendungen verlangen allerdings, dass die Schrift ordnungsgemäss installiert wird und somit auch in der Registrierdatenbank eingetragen wird. Die erreichen Sie mit dem soeben beschriebenen Vorgehen.

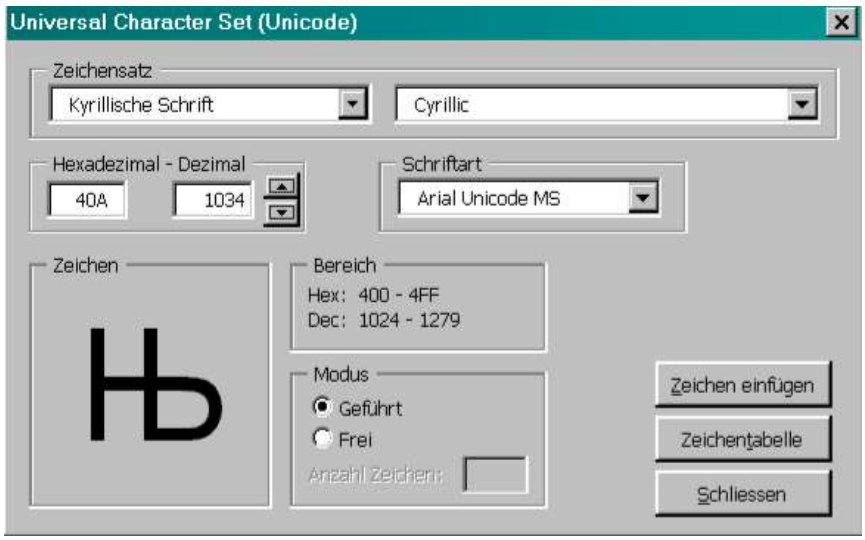
Unicode-Zeichen standardmässig über Word-Dialog eingeben

Wie schon erwähnt steht die Unicode-Unterstützung ausschliesslich in **Word2000** zur Verfügung. Gehen Sie im einzelnen wie folgt vor:

1. Vom Word-Hauptmenü wählen Sie ***Einfügen - Symbol...***.
2. Stellen Sie sicher, dass Sie sich im Reiter **Symbole** befinden und dass im Kombinationsfeld mit der Bezeichnung *Schriftart:* der Wert **Normaler Text** eingestellt ist.
3. Im Kombinationsfeld mit der Bezeichnung *Subset:* können Sie nun die gewünschte Zeichenfamilie einstellen.
4. Markieren Sie alsbald das gewünschte Zeichen und betätigen Sie die Schaltfläche **Einfügen**. Der Dialog bleibt dabei offen, ist also nicht-modaler Natur (d.h. Sie können sich im Dokument bewegen und arbeiten während der Dialog offen bleibt) und Sie können auf Wunsch das nächste Zeichen einfügen.

VBA-Routine zur Darstellung der Unicode-Familien

Nachfolgend, wie versprochen, eine VBA-Routine, welche es Ihnen erlaubt, in Word eine Tabelle für eine beliebige Unicode-Zeichenfamilie zu erstellen. Die Routine kann auch dazu gebraucht werden, ein Unicode-Zeichen nach Wunsch in das Dokument einzubringen.



Implementierung der Lösung

Um die folgende Lösung auf Ihrem Rechner verfügbar zu machen, gehen Sie wie folgt vor:

1. Laden Sie die folgende kleine Datei herunter, welche die Definitionen sowohl für die Unicode-Familien wie auch die Definitionen für die Schaltflächen auf der UserForm beinhaltet, welche Sie benötigen.Unicode Initialisierungsdatei.
2. Suchen Sie auf Ihrem Rechner nach der Datei mit dem Namen *winword.exe* und merken Sie sich das Verzeichnis in welcher sich diese Datei befindet.
3. Kopieren Sie die herunter geladene Datei in dieses Verzeichnis.
4. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** (Rechtsklick im Windows-Explorer - Kontextmenü - Öffnen).
5. Wechseln Sie mit der Tastenkombination **[Strg + F11]** in die VBA-Umgebung.
6. Wählen Sie im VBA-Editor im Hauptmenü den Dialog ***Einfügen - Modul*** und kopieren Sie den kurzen Code (3-Zeiler), welcher unten abgebildet ist, in das leere Code-Fenster.
7. Wählen Sie nun im Hauptmenü den Dialog ***Einfügen - UserForm***.
8. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt **Code anzeigen**.
9. Löschen Sie nun die zwei Code-Zeilen, welche von Word automatisch generiert wurden und kopieren Sie den grösseren Code, welcher unten abgebildet ist in das nun leere Fenster.
10. Lassen Sie sich nun mit der Taste **F4** den Eigenschaftsdialog der UserForm anzeigen.
11. Ändern Sie nun die Eigenschaft **Name** von *UserFormx* nach **frmUnicode**..
12. Speichern Sie nun die Vorlage und verlassen Sie Word.
13. Starten Sie Word wieder, und rufen Sie über ***Extras - Makro - Makros...*** das Makro mit dem Namen **UnicodeStart** auf.

```
Sub UniCodeStart()
  frmUnicode.Show
End Sub
```

(Ruft die UserForm zur Erstellung einer Uncode-Tabelle auf.)

```
' Unicode Tabelle
' Erstellet von René Probst / eku / CH-8152 Glattbrugg
' am 08. Juli 2000
'
Private Const Titel = "Universal Character Set (Unicode)"
Private Const INIDatei = "VBA-Projekte-UCS.ini"
Private AnzSprachen As Long
Private AnzFamilien As Long
Private x() As Variant
Private y(9) As String
Private WithEvents cbSprachen As MSForms.ComboBox
Private WithEvents cbFamilien As MSForms.ComboBox
Private WithEvents cbSchriften As MSForms.ComboBox
Private WithEvents tbHex As MSForms.TextBox
Private WithEvents tbDec As MSForms.TextBox
Private WithEvents spnDec As MSForms.SpinButton
```

```
Private WithEvents obAuto As MSForms.OptionButton
Private WithEvents obFrei As MSForms.OptionButton
Private WithEvents tbAnzahl As MSForms.TextBox
Private WithEvents cmdEinfügen As MSForms.CommandButton
Private WithEvents cmdTabelle As MSForms.CommandButton
Private WithEvents cmdSchliessen As MSForms.CommandButton
Private lblBereich As MSForms.Label
Private lblView As MSForms.Label
Private lblAnzahl As MSForms.Label
Private lblMsg As MSForms.Label
Public docUCS As Document

Private Sub UserForm_Initialize()
  Me.Caption = Titel
  Me.Height = 207
  Me.Width = 351
  Dim IniData As String
  IniData = IniDateiLaden
  IniData = FormdatenLaden(IniData)
  ZeichensatzdatenLaden IniData
  IniData = ""
  cbSprachen.ListIndex = 0
  SchriftenLaden
  lblMsg.ForeColor = &H80000002
  tbHex.MaxLength = 4
  tbDec.MaxLength = 5
  lblView.Font.Size = 60
  obAuto.Value = True
  cmdSchliessen.Cancel = True
  cmdEinfügen.Default = True
  cbSprachen.Style = fmStyleDropDownList
  cbFamilien.Style = fmStyleDropDownList
  cbSchriften.Style = fmStyleDropDownList
End Sub

Private Function IniDateiLaden() As String
  Close
  Pfad = Options.DefaultFilePath(wdProgramPath) & "\" & INIDatei
  chk = Dir(Pfad)
  If chk = "" Then
    MsgBox "Die Datei " & Chr(34) & Pfad & Chr(34) & _
       " ist nicht vorhanden.", vbCritical, Titel
    End
  End If
  On Error Resume Next
  Open Pfad For Binary As #1
  If Err.Number > 0 Then
    MsgBox Err.Description, vbCritical, Titel
    End
  End If
  IniDateiLaden = Input(LOF(1), #1)
End Function
```

```
Private Function FormdatenLaden(tStr As String)
  Dim C As Object, D As MSForms.Control
  i = 0
  ofs = InStr(tStr, Chr(8))
  While ofs > 0
    i = i + 1
    tmp = Left(tStr, ofs - 1)
    If tmp <> "-" Then
      Select Case i Mod 10
        Case 1: Set C = GetControl(tmp)
        Case 2: p1 = tmp
        Case 3
          Set D = C.Controls.Add("Forms." & p1 & ".1", tmp, True)
          Select Case tmp
            Case "cbSprachen": Set cbSprachen = D
            Case "cbFamilien": Set cbFamilien = D
            Case "cbSchriften": Set cbSchriften = D
            Case "tbHex": Set tbHex = D
            Case "tbDec": Set tbDec = D
            Case "spnDec": Set spnDec = D
            Case "obAuto": Set obAuto = D
            Case "obFrei": Set obFrei = D
            Case "lblAnzahl": Set lblAnzahl = D
            Case "tbAnzahl": Set tbAnzahl = D
            Case "cmdEinfügen": Set cmdEinfügen = D
            Case "cmdTabelle": Set cmdTabelle = D
            Case "cmdSchliessen": Set cmdSchliessen = D
            Case "lblBereich": Set lblBereich = D
            Case "lblView": Set lblView = D
            Case "lblMsg": Set lblMsg = D
          End Select
        Case 4: D.Top = tmp
        Case 5: D.Left = tmp
        Case 6: D.Height = tmp
        Case 7: D.Width = tmp
        Case 8: D.Caption = tmp
        Case 9: D.Accelerator = tmp
        Case 0: D.TextAlign = tmp
      End Select
    End If
    tStr = Right(tStr, Len(tStr) - ofs)
    ofs = InStr(tStr, Chr(8))
  Wend
  FormdatenLaden = tStr
End Function

Private Sub ZeichensatzdatenLaden(tStr As String)
  Dim z(2) As Long
  For i = 0 To 2
    z(i) = Left(tStr, InStr(tStr, Chr(0)))
    tStr = Right(tStr, Len(tStr) - InStr(tStr, Chr(0)))
  Next i
  ReDim x(z(0) - 1, z(1), z(2) - 1)
```

```
  i = -1
  While Len(tStr) > 3
    ofs1 = InStr(tStr, Chr(255))
    ofs2 = InStr(ofs1 + 2, tStr, Chr(255))
    Select Case Mid(tStr, ofs1 + 1, 1)
      Case Chr(1)
        i = i + 1
        x(i, 0, 0) = Mid(tStr, ofs1 + 2, ofs2 - ofs1 - 2)
        cbSprachen.AddItem x(i, 0, 0)
        j = 0
      Case Chr(2)
        j = j + 1
        x(i, j, 0) = Mid(tStr, ofs1 + 2, ofs2 - ofs1 - 2)
        k = 0
      Case Chr(3), Chr(4), Chr(5)
        k = k + 1
        x(i, j, k) = Mid(tStr, ofs1 + 2, ofs2 - ofs1 - 2)
    End Select
    tStr = Right(tStr, Len(tStr) - ofs2 + 1)
  Wend
  Close #1
   If Err.Number > 0 Then
    MsgBox "Die Datei " & Chr(34) & Pfad & Chr(34) & _
       " hat ein ungültiges Format.", vbCritical, Titel
    End
  End If
  On Error GoTo 0
End Sub

Private Sub SchriftenLaden()
  y(0) = "Andale Mono"
  y(1) = "Trebuchet MS"
  y(2) = "Georgia"
  y(3) = "Verdana"
  y(4) = "Comic Sans MS"
  y(5) = "Arial"
  y(6) = "Impact"
  y(7) = "Times New Roman"
  y(8) = "Courier"
  y(9) = "Arial Unicode MS"
  Dim z() As String
  i = -1
  For Each oFont In Application.FontNames
    For j = 0 To 9
      If y(j) = oFont Then
        i = i + 1
        ReDim Preserve z(i)
        z(i) = oFont
        Exit For
      End If
    Next j
  Next
  WordBasic.SortArray z()
```

```
  cbSchriften.List = z()
  Flag = False
  For i = 0 To UBound(z)
    If z(i) = "Arial Unicode MS" Then
      cbSchriften.ListIndex = i
      Flag = True
      Exit For
    End If
  Next i
  If Flag = False Then
    For i = 0 To UBound(z)
      If z(i) = "Arial" Then
        cbSchriften.ListIndex = i
        Flag = True
        Exit For
      End If
    Next i
  End If
  If Flag = False Then cbSchriften.ListIndex = 0
End Sub

Private Sub cbSprachen_Change()
  If cbSprachen.ListIndex = -1 Then Exit Sub
  cbFamilien.Clear
  For i = 1 To 255
    If x(cbSprachen.ListIndex, i, 0) = "" Then Exit For
    cbFamilien.AddItem x(cbSprachen.ListIndex, i, 0)
  Next i
  cbFamilien.ListIndex = 0
End Sub

Private Sub cbFamilien_Change()
  lblMsg.Caption = ""
  If cbFamilien.ListIndex = -1 Then Exit Sub
  spnDec.Min = x(cbSprachen.ListIndex, cbFamilien.ListIndex + 1, 1)
  spnDec.Max = x(cbSprachen.ListIndex, cbFamilien.ListIndex + 1, 2)
  tbDec.Text = x(cbSprachen.ListIndex, cbFamilien.ListIndex + 1, 1)
  obAuto.Value = True
  CodeChange
End Sub

Private Sub cbSchriften_Change()
  lblMsg.Caption = ""
  lblView.Font = cbSchriften.Value
End Sub

Private Sub spnDec_SpinUp()
  lblMsg.Caption = ""
  If spnDec.Value > spnDec.Max Then Exit Sub
  tbDec.Text = spnDec.Value
  CodeChange
End Sub
```

```
Private Sub spnDec_SpinDown()
  lblMsg.Caption = ""
  If spnDec.Value < spnDec.Min Then Exit Sub
  tbDec.Text = spnDec.Value
  CodeChange
End Sub

Private Sub tbHex_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
  lblMsg.Caption = ""
  If tbHex.Locked = True Then
    Beep
    lblMsg = _
       "Schalten Sie in den freien Modus um diesen Wert zu ändern."
    Exit Sub
  End If
  a = Asc(UCase(Chr(KeyAscii)))
  If a < 48 Or a > 57 And a < 65 Or a > 70 Then
    Beep
    KeyAscii = 0
  Else
    KeyAscii = a
  End If
End Sub

Private Sub tbDec_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
  lblMsg.Caption = ""
  If tbDec.Locked = True Then
    Beep
    lblMsg = _
       "Schalten Sie in den freien Modus um diesen Wert zu ändern."
    Exit Sub
  End If
  If KeyAscii < 48 Or KeyAscii > 57 Then
    Beep
    KeyAscii = 0
  End If
End Sub

Private Sub tbDec_Change()
  If tbDec.Text = "" Then
    tbHex.Text = ""
    Exit Sub
  End If
  If Val(tbDec.Text) > 65535 Then
    Beep
    tbDec.Text = spnDec.Value
    tbDec.SelStart = 0
    tbDec.SelLength = Len(tbDec.Text)
    lblMsg.Caption = "Dieser Wert darf höchstens 65535 betragen."
    Exit Sub
  End If
  CodeChange
End Sub
```

```
Private Sub tbHex_Change()
  If tbHex.Text = "" Then
    tbDec.Text = ""
    Exit Sub
  End If
  tbDec.Text = Dec(tbHex.Text)
End Sub

Private Sub tbAnzahl_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
  lblMsg.Caption = ""
  If KeyAscii < 48 Or KeyAscii > 57 Then
    Beep
    KeyAscii = 0
  End If
End Sub

Private Sub tbAnzahl_Change()
  CodeChange
End Sub

Private Sub obAuto_Click()
  tbDec.Locked = True
  tbHex.Locked = True
  tbAnzahl.Enabled = False
  tbAnzahl.BackColor = &H8000000F
  lblAnzahl.Enabled = False
  cbFamilien_Change
End Sub

Private Sub obFrei_Click()
  lblMsg.Caption = ""
  tbDec.Locked = False
  tbHex.Locked = False
  tbAnzahl.Enabled = True
  tbAnzahl.BackColor = &H80000005
  lblAnzahl.Enabled = True
  If tbAnzahl.Text = "" Then tbAnzahl.Text = 256
  tbAnzahl.MaxLength = 4
  spnDec.Min = 0
  spnDec.Max = 65535
  tbDec.SetFocus
  tbDec.SelStart = 0
  tbDec.SelLength = Len(tbDec.Text)
  CodeChange
End Sub

Private Sub cmdEinfügen_Click()
  lblMsg.Caption = ""
  If tbDec.Text = "" Then
    Beep
    tbDec.SetFocus
    lblMsg.Caption = "Fehlender Wert."
```

```
    Exit Sub
  End If
  cmdSchliessen.Enabled = False
  ThisDocument.Activate
  Selection.Font.Name = cbSchriften.Value
  Application.ScreenRefresh
  Selection.TypeText Text:=ChrW(tbDec.Text)
  Application.ScreenRefresh
  cmdSchliessen.Enabled = True
End Sub

Private Sub cmdTabelle_Click()
  lblMsg.Caption = ""
  If obFrei.Value = True Then
    If Val(tbDec.Text) < 32 Then
      Beep
      lblMsg.Caption = "Dieser Wert darf nicht unter 32 liegen."
      tbDec.SetFocus
      tbDec.Text = spnDec.Value
      tbDec.SelStart = 0
      tbDec.SelLength = Len(tbDec.Text)
      Exit Sub
    End If
    If tbAnzahl.Text = "0" Or tbAnzahl.Text = "" Then
      Beep
      tbAnzahl.SetFocus
      lblMsg.Caption = "Dieser Wert muss zwischen 1 und 9999 liegen."
      tbAnzahl.SelStart = 0
      tbAnzahl.SelLength = Len(tbAnzahl.Text)
      Exit Sub
    End If
  End If
  cmdEinfügen.Enabled = False
  cmdSchliessen.Enabled = False
  DoEvents
  On Error Resume Next
  docUCS.Activate
  Fehler = Err.Number
  On Error GoTo 0
  If Fehler > 0 Then DokumentEinrichten
  If obAuto.Value = True Then KopfinformationErstellen
  TabelleErstellen
  Application.ScreenUpdating = True
  cmdEinfügen.Enabled = True
  cmdSchliessen.Enabled = True
  cmdSchliessen.SetFocus
  lblMsg.Caption = "Die Tabelle wurde erstellt."
  ThisDocument.Activate
End Sub

Private Sub cmdSchliessen_Click()
  Unload Me
End Sub
```

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If cmdSchliessen.Enabled = False Then Cancel = True
End Sub

Private Sub DokumentEinrichten()
  Dim docUCS As Document, FV As Style
  lblMsg.Caption = "Das Dokument wird eingerichtet."
  Set docUCS = Documents.Add
  Application.ScreenUpdating = False
  DoEvents
  ActiveWindow.ActivePane.View.ShowAll = False
  On Error Resume Next
  docUCS.Styles("UCS Absatz").Delete
  docUCS.Styles("UCS dots").Delete
  docUCS.Styles("UCS Hex").Delete
  docUCS.Styles("UCS Dec").Delete
  docUCS.Styles("UCS Char").Delete
  On Error GoTo 0
  Set FV = docUCS.Styles.Add(Name:="UCS Absatz", Type:=1)
  FV.ParagraphFormat.SpaceBefore = 3
  FV.ParagraphFormat.SpaceAfter = 3
  FV.ParagraphFormat.Alignment = wdAlignParagraphJustify
  Set FV = ActiveDocument.Styles.Add(Name:="UCS dots", Type:=1)
  FV.BaseStyle = "UCS Absatz"
  docUCS.Styles("UCS dots").LinkToListTemplate ListTemplate:= _
    ListGalleries(1).ListTemplates(1), ListLevelNumber:=1
  Set FV = docUCS.Styles("Standard")
  FV.ParagraphFormat.SpaceBefore = 2
  FV.ParagraphFormat.SpaceAfter = 2
  Set FV = ActiveDocument.Styles.Add(Name:="UCS Hex", Type:=1)
  FV.ParagraphFormat.Alignment = wdAlignParagraphCenter
  FV.Font.Name = "Times New Roman"
  FV.Font.Size = 9
  FV.Font.ColorIndex = wdDarkRed
  Set FV = ActiveDocument.Styles.Add(Name:="UCS Dec", Type:=1)
  FV.ParagraphFormat.Alignment = wdAlignParagraphRight
  FV.Font.Name = "Times New Roman"
  FV.Font.Size = 9
  FV.Font.ColorIndex = wdBlue
  Set FV = ActiveDocument.Styles.Add(Name:="UCS Char", Type:=1)
  FV.ParagraphFormat.Alignment = wdAlignParagraphCenter
  FV.Font.Name = cbSchriften.Value
  FV.Font.Size = 14
End Sub

Private Sub KopfinformationErstellen()
  Dim oRange As Range
  Set docUCS = ActiveDocument
  lblMsg.Caption = "Die Kopfinformationen werden erstellt."
  Application.ScreenUpdating = False
  DoEvents
  Set oRange = docUCS.Paragraphs.Last.Range
```

```
  Pos = oRange.Start
  oRange.Style = wdStyleHeading3
  oRange.ParagraphFormat.SpaceBefore = 15
  oRange.InsertAfter cbFamilien.Value & " " & Chr(250) & "U" & _
     HexPadding(spnDec.Min) & " - " & HexPadding(spnDec.Max) & _
        Chr(254) & " Dezimal " & Chr(251) & spnDec.Min & _
           " - " & spnDec.Max & Chr(254) & vbCrLf
  oRange.SetRange Start:=oRange.End, End:=oRange.End
  oRange.Style = "UCS Absatz"
  If x(cbSprachen.ListIndex, cbFamilien.ListIndex + 1, 3) <> "" Then
    oRange.InsertAfter x(cbSprachen.ListIndex, cbFamilien.ListIndex + 1, 3)
    oRange.InsertAfter vbCrLf
  End If
  oRange.SetRange Start:=Pos, End:=docUCS.Range.End
  With oRange.Find
    .ClearFormatting
    .Format = True
    .MatchWildcards = True
    .Text = Chr(247) & "*" & Chr(254)
    .Replacement.ClearFormatting
    .Replacement.Font.Bold = True
    .Replacement.Font.Italic = True
    .Execute Replace:=wdReplaceAll
    .Text = Chr(248) & "*" & Chr(254)
    .Replacement.ClearFormatting
    .Replacement.Font.Italic = True
    .Execute Replace:=wdReplaceAll
    .Text = Chr(249) & "*" & Chr(254)
    .Replacement.ClearFormatting
    .Replacement.Font.Bold = True
    .Execute Replace:=wdReplaceAll
    .Text = Chr(250) & "*" & Chr(254)
    .Replacement.ClearFormatting
    .Replacement.Font.ColorIndex = wdDarkRed
    .Execute Replace:=wdReplaceAll
    .Text = Chr(251) & "*" & Chr(254)
    .Replacement.ClearFormatting
    .Replacement.Font.ColorIndex = wdBlue
    .Execute Replace:=wdReplaceAll
    .Text = Chr(253) & "*" & Chr(254)
    .Replacement.ClearFormatting
    .Replacement.Style = "UCS dots"
    .Execute Replace:=wdReplaceAll
    .Format = Flase
    .MatchWildcards = False
    For i = 247 To 254
      If i <> 252 Then .Execute FindText:=Chr(i), ReplaceWith:="", Replace:=2
    Next i
  End With
End Sub

Private Sub TabelleErstellen()
  Dim oTable As Table, oZelle As Cell, oRange As Range
```

```
Set docUCS = ActiveDocument
Set oRange = docUCS.Paragraphs.Last.Range
lblMsg.Caption = "Die Tabelle wird nun erstellt."
Application.ScreenUpdating = False
DoEvents
Application.ScreenUpdating = False
Dim sStart As Long, sStop As Long, hStart As Long, dStart As Long
sStart = CLng(tbDec.Text)
If obAuto.Value = True Then
  hStart = spnDec.Min - 1
  dStart = hStart
  sStop = spnDec.Max
  AZ = Int((spnDec.Max - spnDec.Min) / 10)
  If Int((spnDec.Max - spnDec.Min) Mod 10) > 0 Then AZ = AZ + 1
Else
  hStart = CLng(tbDec.Text) - 1
  dStart = hStart
  sStop = CLng(tbDec.Text) + CLng(tbAnzahl.Text) - 1
  AZ = Int((tbAnzahl.Text) / 10)
  If Int((tbAnzahl.Text) Mod 10) > 0 Then AZ = AZ + 1
  oRange.InsertAfter vbCrLf
End If
oRange.Paragraphs.Last.Previous.Range.ParagraphFormat.SpaceAfter = 12
Set oTable = docUCS.Tables.Add(Range:=oRange, NumRows:=2, _
   NumColumns:=20, DefaultTableBehavior:=wdWord8TableBehavior)
oTable.AutoFormat Format:=wdTableFormat3DEffects3, _
  ApplyBorders:=True, ApplyShading:=True, ApplyFont:=True, _
     ApplyColor:=True, ApplyHeadingRows:=False, ApplyLastRow:=False, _
       ApplyFirstColumn:=False, ApplyLastColumn:=False, AutoFit:=False
oTable.AutoFitBehavior wdAutoFitFixed
For i = 2 To oTable.Columns.Count Step 2
  oTable.Columns(i).Cells.Merge
Next i
For Each oZelle In oTable.Range.Cells
  If oZelle.ColumnIndex Mod 2 = 0 Then
    oZelle.Range.Style = "UCS Char"
  Else
    If oZelle.RowIndex = 1 Then
      oZelle.Range.Style = "UCS Hex"
    Else
      oZelle.Range.Style = "UCS Dec"
    End If
  End If
Next
Set oRange = oTable.Range
While oTable.Rows.Count < AZ * 2
  oRange.SetRange Start:=oRange.End + 2, End:=oRange.End + 2
  oRange.FormattedText = oTable.Range.FormattedText
Wend
If oTable.Rows.Count > AZ * 2 Then
  docUCS.Range(Start:=oTable.Range.Cells(AZ * 30 + 1).Range.Start, _
     End:=oTable.Range.End).Rows.Delete
End If
```

```
  For Each oZelle In oTable.Range.Cells
    If oZelle.ColumnIndex Mod 2 = 1 Then
      If oZelle.RowIndex Mod 2 = 1 Then
        hStart = hStart + 1
        tmp = Hex(hStart)
        While Len(tmp) < 4
          tmp = "0" & tmp
        Wend
        If hStart <= sStop Then
          oZelle.Range.Text = tmp
          tbDec.Text = hStart
          CodeChange
          DoEvents
        End If
      Else
        dStart = dStart + 1
        If dStart <= sStop Then oZelle.Range.Text = dStart
      End If
    Else
      If Len(oZelle.Range.Text) = 2 And hStart <= sStop Then
        oZelle.Range.Text = ChrW(hStart)
      End If
    End If
  Next
  oTable.AutoFitBehavior wdAutoFitContent
  docUCS.Range.InsertAfter vbCrLf
  If obAuto.Value = False Then
    If docUCS.Tables.Count = 1 Then Selection.SplitTable
  End If
  tbDec.Text = sStart
  CodeChange
  DoEvents
End Sub

Private Sub CodeChange()
  spnDec.Value = tbDec.Text
  tbHex.Text = Hex(tbDec.Text)
  lblView.Caption = ""
  lblView.Caption = ChrW(tbDec.Text)
  If obAuto.Value = True Then
    lblBereich.Caption = "Hex:  " & HexPadding(spnDec.Min) & _
        " - " & HexPadding(spnDec.Max) & vbCrLf & _
           "Dec:  " & spnDec.Min & " - " & spnDec.Max
  Else
    If tbAnzahl.Text = "0" Or tbAnzahl.Text = "" Then
      lblBereich.Caption = "Hex:  " & HexPadding(tbDec.Text) & _
        " - " & vbCrLf & "Dec:  " & tbDec.Text & " - "
    Else
      lblBereich.Caption = "Hex:  " & HexPadding(tbDec.Text) & _
        " - " & HexPadding(Val(tbDec.Text) + Val(tbAnzahl.Text) - 1) _
           & vbCrLf & "Dec:  " & tbDec.Text & " - " & _
              Val(tbDec.Text) + Val(tbAnzahl.Text) - 1
    End If
```

```
  End If
End Sub

Private Function Dec(D As String) As Long
  t = "0123456789ABCDEF"
  For i = Len(D) To 1 Step -1
    Dec = Dec + (InStr(t, Mid(D, i, 1)) - 1) * 16 ^ (Len(D) - i)
  Next i
End Function

Private Function HexPadding(D As Long) As String
  HexPadding = Hex(D)
  While Len(HexPadding) < 4
    HexPadding = "0" & HexPadding
  Wend
  HexPadding = "U" & HexPadding
End Function

Private Function GetControl(ByVal Ctrl As String) As Object
  If Ctrl = "Me" Then
    Set GetControl = Me
  Else
    For Each GetControl In Me.Controls
      If GetControl.Name = Ctrl Then Exit Function
    Next
  End If
End Function
```

(Dieser Code für die UserForm ermöglicht es, sowohl ein Unicode-Zeichen in ein Dokument einzufügen sowie auch eine Tabelle von einer Unicode-Familie zu erstellen.)

Problembehandlung im Zusammenhang mit der Darstellung der Unicode-Zeichen

Während ich diese Arbeit zusammengestellt habe, bin ich leider auf Einschränkungen verschiedenster Natur gestossen. Es scheint, dass das *Universal Character Set*, obwohl schon in der 3. Version verfügbar und obwohl sich die meisten relevanten Hersteller von Software dem Konzept anschliessen wollen, doch noch einiges an Kopfzerbrechen hervorruft.

Sollten Sie bei der Anwendung des hier bereitgestellten Materials auf Probleme stossen, gehen Sie bitte systematisch folgende Punkte durch.

- Sollten Sie in MS Word nur sehr wenige der Schriftfamilien darstellen können, bei der Mehrzahl der Familien erscheinen die Zeichen aber nur als Kästchen, dann sollten Sie wie oben beschrieben die neusten Unicode-fähigen Schriften herunter laden.
- Sollten Sie in MS Word nur etwa die Hälfte der Schriftfamilien darstellen können, dann ist das so lange als normal zu betrachten, wie Ihr Rechner nicht über die Schrift mit dem Namen **Arial Unicode MS** verfügt. Falls Sie mit **Word2000** arbeiten, können Sie diese Schrift. wie weiter oben erklärt. von der Office2000-CD nachinstallieren.
- Sollten Sie im bereit gestellten PDF-Dokument statt echten Zeichen eine Grafik der Zeichentabelle sehen, dann liegt es daran, das die Schrift *Arial Unicode MS* auf dem Stand Unicode Version 2.0 ist, das Unicode-Konsortium aber bereits die dritte Version veröffentlicht hat und jetzt an der vierten Version arbeitet.
- Sollten Sie im bereit gestellten PDF-Dokument eine Zeichentabelle sehen, welche **ausschliesslich** mit Kästchen gefüllt ist, dann liegt es daran, dass das PDF-Format diese nicht übernehmen kann bzw. anzeigen kann. Word2000 kann dies aber und auch eine Tabelle, welche Sie mit der hier vorgestellten VBA-Routine erstellen, wird diese Zeichen in Word2000 anzeigen.
- Sollten Sie im bereit gestellten PDF-Dokument eine Zeichentabelle sehen, welche **teilweise** mit Kästchen gefüllt ist, dann handelt es sich um Platzhalter, also Zeichen, welche vom Unicode-Konsortium (noch) nicht vergeben wurden. Diese Erscheinung ist normal. Es gibt da nichts darstellbares, auch nicht in MS Word.
- Sollten Sie planen, selbst eine solche Zeichentabelle mit dem *Adobe Acrobat 4.0 Distiller*, zu exportieren, dann wird es Sie vielleicht interessieren, dass wenn Sie Unicode Zeichen im relativ kleinen Bereich von *FB00 bis FB4F* darstellen wollen und zwar formatiert mit der Schriftart *Arial Unicode MS*, dann wird der Acrobat-Treiber MS Word mit einer Schutzverletzung zum abschmieren bringen.

Links zu diesem Thema

Eigene Links

► **Unicode Tabellen (Universal Character Set) - PDF Format (3.4 MB)**

► **Wissenswertes über die Zeichensatztabellensuppe**

Microsoft

► **Unicode taugliche TrueType Schriften**

Das Unicode Konsortium

► **Unicode Home Page**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Unicode.htm

René Probst, Juli 2000

Änderungen

- 9. Februer 2001 - Überarbeitet

Office2000: Wie kann ich die Word-Dialoge *Datei - öffnen*, *Datei - speichern unter*, *Grafik - einfügen*, etc. anpassen?

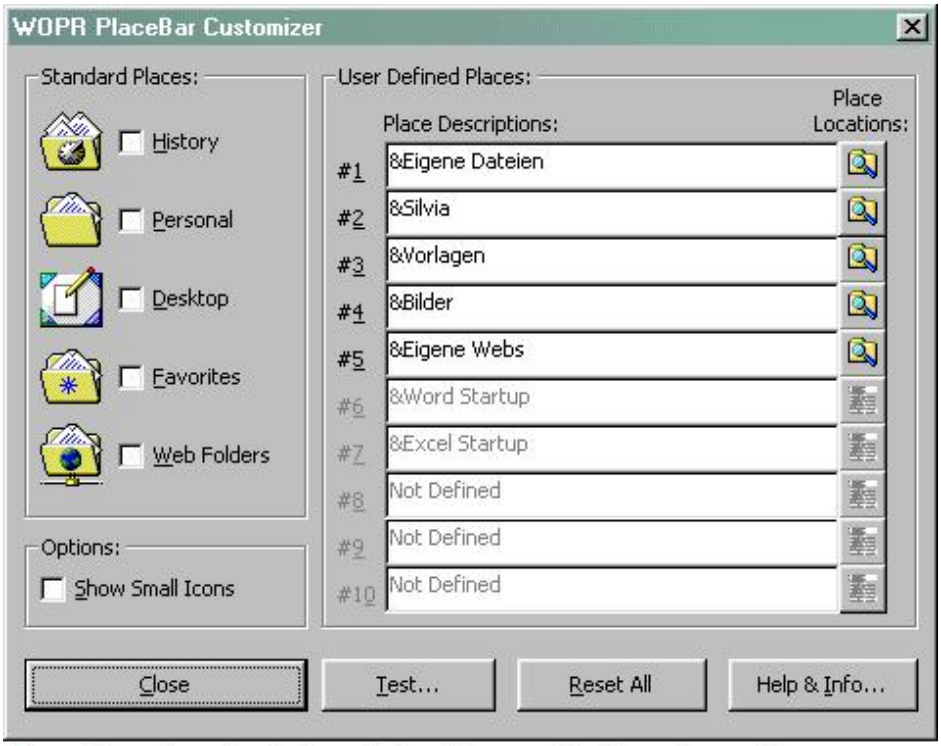
◄ Zurück zum Hauptindex

Frage

> **Wenn ich in einer Office2000 Anwendung den Dialog *Datei - Öffnen* aufrufe, dann erkenne ich links eine Leiste mit grossen Symbolen, welche es mir auf bequeme Art und Weise erlaubt, zwischen verschiedenen Verzeichnissen zu wechseln. Die Auswahl dieser Verzeichnisse entspricht aber nicht ganz meinem Geschmack, deshalb möchte ich diese Leiste anpassen.**

Antwort

Sie können im Internet ein kleines Tool beziehen, welches sich genau dieser Sache annimmt. Das Feature heisst **WOPR PlaceBar Customizer**, wird auch von Microsoft empfohlen und ermöglicht es, diese Leiste anzupassen. Sie dürfen dieses Tool ohne Einschränkungen gratis benutzen. Damit es Ihnen zur Verfügung steht, gehen Sie wie folgt vor:



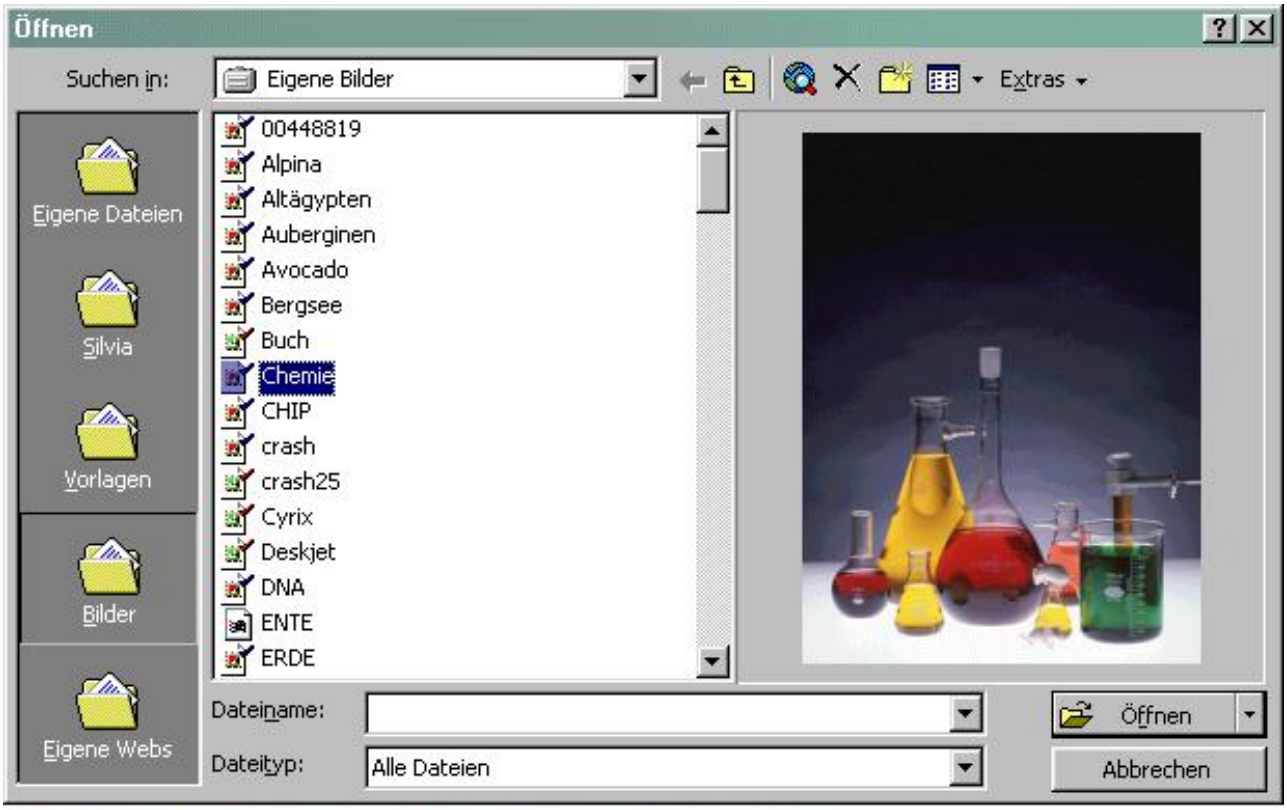
Über diesen beschaulichen Dialog, können Sie die Ordner-Leiste anpassen.
Je nach dem, ob Sie grosse Symbole wünschen (wie im Bild) oder aber kleine Symbole, stehen Ihnen dann 5 bzw. 10 Einträge nach Ihrem Wunsch zur Verfügung.

Place Bar Customizer einrichten

Die Schritte um dieses Tools einzurichten, sind sehr einfach. Laden Sie die Installationsroutine herunter, entpacken sie die Datei und Installieren Sie das Tool mittels Doppelklick. Gehen Sie im einzelnen wie folgt vor:

1. Besuchen Sie den spezifischen ***WOPR*** Link von *Woody Leonhard's*, welcher am Schluss dieses Skriptes eingeblendet ist Der Link führt direkt zum Ziel.
2. Laden Sie den dort vorgestellten *PlaceBar Customizer* herunter. Der Download beträgt knapp 400kb.
3. Entpacken Sie die herunter geladenen Zip-Datei. Sie enthält eine einzige Datei mit dem Namen **PlaceBar.exe**.
4. Schliessen Sie sämtliche Office-Anwendungen und führen Sie einen Doppelklick auf die genannte Installationsdatei aus.
5. Wenn Sie nach erfolgreicher Installation irgend eine Office2000-Anwendung starten, können Sie vom Hauptmenü - jetzt neu - den Dialog ***Extras - WOPR Place Bar Customizer*** aufrufen, und dann die gewünschten Veränderungen bezüglich dieser Leiste ausführen.

Resultat der Anpassung



Das Resultat der Anpassung sieht dann wie oben abgebildet aus.

Anmerkungen

! Die Lösung wurde ausschliesslich für **Office2000** entwickelt.

- Sie können zwischen **grossen** und **kleinen Symbolen** auswählen. Wenn Sie grosse Symbole wählen, passen **fünf** Einträge in die Leiste, wenn Sie dagegen kleine Symbole wählen, passen **zehn** Einträge in die Leiste.

Links zu diesem Thema

Woody Leonhard's

► Woodies Office2000 Place Bar Customizer

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/WOPR.htm

René Probst, November 2000

Änderungen

- --- nil ---

Der Word-Dialog «*Datei - Neu...*» soll statt grossen Symbolen kleine Symbole anzeigen

◄ Zurück zum Hauptindex

Problem

> **Wenn ich in MS Word den Dialog «*Datei - Neu...*» aufrufe, werden mir immer grosse Symbole angezeigt. Dies erweist sich aber in meiner Umgebung als unübersichtlich. Wie kann ich Word dazu zwingen, immer kleine Symbole anzuzeigen?**

Antwort

Sie können den Ablauf, welcher bei *Datei - Neu...* abläuft mit einem kleinen Makro beeinflussen. Im folgenden wird ein kleines Makro vorgestellt, welches Word dazu forciert, die Liste der Dokumentenvorlagen in einer bestimmten Darstellung anzuzeigen.

Lösung

Um die Lösung auf Ihrem Rechner einzurichten, gehen Sie wie folgt vor:

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** (Rechtsklick im Windows-Explorer auf *Normal.dot* und im Kontextmenü **Öffnen** wählen).
2. Wechseln Sie mit der Tastenkombination **[Alt+F11]** in die VBA-Umgebung.
3. Wählen Sie dort vom Hauptmenü den Dialog ***Einfügen - Modul***.
4. Kopieren Sie den folgenden VBA-Code in das leere Fenster, welches sich geöffnet hat.
5. **Speichern** Sie die *Normal.dot* und verlassen Sie Word.
6. Beim nächsten Start von Word, sollte Ihnen die Änderung zur Verfügung stehen.

```
Private Sub FileNew()
  SendKeys "%2"
  Dialogs(wdDialogFileNew).Show
End Sub
```

(Forciert Word dazu, beim Aufruf des Dialoges *Datei - Neu...* bei der Anzeige der Dokumentenvorlagen kleine Symbole zu verwenden.)

Anmerkungen

Um die Lösung auf Ihrem Rechner einzurichten, gehen Sie wie folgt vor:

- Wenn Sie statt «*%2*» **«%1»** codieren, werden forciert grosse Symbole angezeigt (das ist der Standard), wenn Sie dagegen statt «*%1*» **«%3»** codieren werden die Detailinformationen der Dokumentenvorlage angezeigt.
- Die hier besprochenen Änderung wirkt sich auf den Dialog *Datei - Neu...* in Word aus. Auf den Dialog *Start - Neues Office-Dokument* ex Windows-Startmenü oder ex Office-Shortcut-Leiste hat die Änderung aber keine Auswirkung. Sie können dort nicht eingreifen.
- Die *SendKeys*-Methode gilt im Allgemeinen als nicht ganz zuverlässig. Da es sich bei der hier besprochen Lösung aber um einen sehr kleinen Eingriff handelt, dürfte die Verlässlichkeit in der Praxis gegeben sein.
- Der Name des Prozesses, nämlich **FileNew** ist zwingend. Alternativ könnten Sie den Prozess auch *DateiNeu* nennen. Dies wäre aber nicht international und würde ausschliesslich in der deutschsprachigen Word-Version funktionieren.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/DateiNeu.htm

René Probst, November 2000

Änderungen

- --- nil ---

Wie kann ich einen Dialog anzeigen, welcher es dem Benutzer erlaubt, einen Ordner bzw. eine Datei auszuwählen?

◄ Zurück zum Hauptindex

Frage

> Beim Ablauf eines Makros soll dem Benutzer die Möglichkeit geboten werden, einen Ordner bzw. eine Datei, welche sich auf seinem Rechner befindet, auszuwählen. Kann ich dafür auf einen Word-integriereten Dialog zurückgreifen und wenn ja, wie steuere ich dann diesen Dialog? Gibt es Alternativen zu den Word-integrierten Dialogen, Alternativen, welche sich einfach in VBA einbauen lassen?.

Diese Seite

▼ Antwort
▼ Dialog zur Auswahl eines Ordners mittels Word-Dialog
▼ Dialog zur Auswahl einer Datei mittels Word-Dialog
▼ Dialog zur Auswahl eines Ordners oder einer Datei (Windows-API) aus einem Standardmodul
▼ Dialog zur Auswahl eines Ordners oder einer Datei (Windows-API) aus einer UserForm
▼ Links zu diesem Thema

Antwort

▲ Zurück zum Anfang

Zuerst sollten Sie sich entscheiden, ob Sie einen Word-integrierten Dialog verwenden wollen oder gleich ein geeignetes Windows-API (Programmschnittstelle), welches nicht auf MS Office sondern auf bestehende Ressourcen im Windows-Betriebssystem abstellt. Da das API für alle Windows-Versionen verwendet werden kann und der MS Office-Benutzer (fast) nicht um Windows herum kommt, lässt sich auch eine solche Lösung problemlos vertreiben.

Der erste Weg hat den Vorteil, dass Sie weniger Code brauchen um das Ziel zu erreichen. Der Nachteil ist, dass man der Lösung etwas ansieht, dass hier eine Routine zum Einsatz gebracht wird, welche nicht 100%-ig dafür geschaffen wurde.

Beim zweiten Weg kommt mehr Code zum Einsatz. Dafür wird nichts zweckentfremdet und der Benutzer kann kaum etwas machen, was dazu führen könnte, dass Ihre Routine in eine falsche Richtung läuft. Der hier vorgestellte Code zum Aufruf des Windows-API ist stereotyp, stabil und hat sich in der Praxis bewärt.

Dieses Skript geht auf beide Wege ein. Konkret werden Lösungen vorgestellt, um folgende Funktionen zu realisieren:

- Dialog zur Auswahl eines Ordners mittels Word-Dialog.
- Dialog zur Auswahl einer Datei mittels Word-Dialog.
- Dialog zur Auswahl eines Ordners oder einer Datei (Windows-API)

Dialog zur Auswahl eines Ordners mittels Word-Dialog

▲ Zurück zum Anfang

Im folgenden finden Sie den Code, welcher dann zum Zuge kommen kann, wenn der Benutzer einen Ordner aussuchen soll.

Optional können Sie im Kopf des Codes einen Ordner Ihrer Wahl angeben. Dies hat dann zur Folge, dass die Anzeige für den Benutzer mit diesem Ordner beginnt. Möchten Sie dies nicht, dann setzen Sie für die Konstante **Verzeichnis** eine Null-Zeichenfolge (also zwei sich folgende Gänsefüsschen). In diesem Falle entspricht die Vorgabe für den Benutzer dem aktuellen Verzeichnis. Möchten Sie als Vorgabe das Stammverzeichnis (*root*), dann setzen Sie als Wert für die genannte Konstante die Zeichenfolge **"C:\"**.

Sollte der Benutzer einen Ordner ausgewählt haben, dann erhalten Sie den Namen dieses Ordners in einer Variablen mit dem Namen **Ordner** zurück.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private Const Verzeichnis = "C:\Windows"

Sub OrdnerAuswaehlen()
  savEnv = Options.DefaultFilePath(wdCurrentFolderPath)
  On Error Resume Next
  Application.ChangeFileOpenDirectory Verzeichnis
  Fehler = Err.Number
  On Error GoTo 0
  If Fehler > 0 And Fehler <> 4172 Then
    MsgBox Err.Description, vbCritical, "Ordner auswählen"
    Exit Sub
  End If
  With Dialogs(wdDialogCopyFile)
    Select Case .Display
      Case -1 'Der Benutzer hat einen Ordner ausgewählt.
        Ordner = .Directory
        If Left(Ordner, 1) = Chr(34) Then
          Ordner = Mid(Ordner, 2, Len(Ordner) - 3) 'Drop quotes and back slash
        Else
          Ordner = Left(Ordner, Len(Ordner) - 1) 'Drop back slash
        End If
        MsgBox Ordner
      Case 0 'Der Benutzer hat den Vorgang abgebrochen.
        MsgBox "Knacks."
    End Select
  End With
End Sub
```

(Diese Routine lässt den Benutzer einen Ordner auswählen.)

Anmerkungen zum Code

- Sie können den Titel des Dialoges **nicht** Ihren wünschen anpassen. Wenn Ihnen das wichtig erscheint, weichen Sie auf das Windows-API, dessen Beschreibung hier noch folgt, aus.
- Der Benutzer hat **keine** Mehrfachauswahl.

Dialog zur Auswahl einer Datei mittels Word-Dialog

▲ Zurück zum Anfang

Im folgenden finden Sie den Code, welcher dann zum Zuge kommt, wenn der Benutzer eine Datei aussuchen soll.

Optional können Sie im Kopf des Codes einen Ordner Ihrer Wahl angeben. Dies hat dann zur Folge, dass die Anzeige für den Benutzer mit diesem Ordner beginnt. Möchten Sie dies nicht, dann setzen Sie für die Konstante **Verzeichnis** eine Null-Zeichenfolge (also zwei sich folgende Gänsefüsschen). In diesem Falle entspricht die Vorgabe für den Benutzer dem aktuellen Verzeichnis. Möchten Sie als Vorgabe das Stammverzeichnis (*root*), dann setzen Sie als Wert für die genannte Konstante die Zeichenfolge **"C:\"**.

Optional können Sie im Kopf des Codes auch einen Filter angeben. Der Filter **A*** bewirkt dabei, dass nur Dateien deren Namen mit "A" beginnen angezeigt werden. Der Filter ***.htm** bewirkt faktisch, dass ausschliesslich HTML-Dateien angezeigt werden. Schliesslich bewirkt ein Filter wie etwa **A*.htm**, dass ausschliesslich HTML-Dateien deren Namen mit einem "A" beginnen, zur Auswahl stehen.

Sollte der Benutzer eine Datei ausgewählt haben, dann erhalten Sie den Namen dieser Datei in einer Variablen mit dem Namen **Datei** zurück.

```
Private Const Verzeichnis = "C:\Eigene Dateien"
Private Const Filter = "*.doc"

Sub DateiAuswaehlen()
  savEnv = Options.DefaultFilePath(wdCurrentFolderPath)
  On Error Resume Next
  Application.ChangeFileOpenDirectory Verzeichnis
  Fehler = Err.Number
  On Error GoTo 0
  If Fehler > 0 And Fehler <> 4172 Then
    MsgBox Err.Description, vbCritical, "Datei auswählen"
    Exit Sub
  End If
  With Dialogs(wdDialogFileOpen)
    .Name = Filter
    On Error Resume Next
    Dim Antwort As Boolean
    Antwort = .Display
    While Err.Number = 5174
      MsgBox "Sie dürfen nur eine Datei auswählen.", _
         vbExclamation, "Datei auswählen"
      Err.Clear
      Antwort = .Display
    Wend
    Select Case Antwort
      Case -1
        Datei = CurDir & "\" & .Name
        MsgBox Datei, vbInformation, "Datei auswählen."
      Case 0
        MsgBox "Knacks."
    End Select
  End With
  On Error Resume Next
  Application.ChangeFileOpenDirectory savEnv
End Sub
```

(Diese Routine lässt den Benutzer eine Datei auswählen.)

Anmerkungen zum Code

- Sie können den Titel des Dialoges nicht Ihren Wünschen anpassen. Wenn Ihnen das wichtig erscheint, weichen Sie auf das Windows-API, dessen Beschreibung hier noch folgt, aus.
- Sie können den Wert, welcher im Kombinationsfeld mit der Bezeichnung **Dateityp** erscheint **nicht** direkt beeinflussen. Sie können aber im Kopf des Codes einen Filter setzen. Damit ist die Beschränkung auf einen gewissen Dateityp letztlich doch möglich.
- Der Benutzer hat **keine** Mehrfachauswahl. Versucht er es trotzdem, erhält er einen entsprechenden Hinweis und der Dialog wird neu angezeigt.

Dialog zur Auswahl eines Ordners oder einer Datei (Windows-API) aus einem Standardmodul

▲ Zurück zum Anfang

Im folgenden wird die Windows-API Funktion angeboten, welche es erlaubt, wahlweise einen Ordner oder eine Datei auszulesen. Es liegt an Ihnen, vorzugeben, ob der Benutzer einen Ordner bzw. eine Datei auslesen soll.

Sie können im weiteren festlegen, ob der Dialog modaler Natur sein soll. Modal bedeutet, dass der Dialog gebunden ist. Der Benutzer muss dann zwingend eine Auswahl treffen, oder aber den Dialog abbrechen, bevor er in der selben Anwendung weiter arbeiten kann. Anders der ungebundene Zustand. Bei diesem Zustand kann der Dialog den Fokus verlieren (in den Hintergrund treten) und später den Fokus wieder erhalten, dann nämlich, wenn der Benutzer den Dialog anklickt. Im Zweifel wählen Sie hier das modale Verhalten.

Sie können dem Dialog einen Hinweis mitgeben, welcher dann im oberen Bereich des Dialoges wort-wörtlich erscheint.

Falls Sie das Ziel verfolgen, den Dialog aus einer UserForm aufzurufen, dann sollten Sie sich das nächste Kapitel anschauen. Die Bedeutung der Konstanten im Kopf des Codes ist wie folgt:

Bevor Sie beginnen

Sie können die VBA-Routine Ihren eigenen spezifischen Bedürfnissen anpassen. Dazu passen Sie im Kopf des Codes folgende Konstanten an:

- Konstante ***Modus***: Setzen Sie den Wert dieser Konstante wie folgt:
  - **0** Gibt dem Benutzer die Möglichkeit einen **Ordner** auszuwählen.
  - **1** Gibt dem Benutzer die Möglichkeit eine **Datei** auszuwählen..
- Konstante ***Modal***: Setzen Sie den Wert dieser Konstante wie folgt:
  - **0** Der Dialog wird **ungebunden** angezeigt.
  - **1** Der Dialog wird **gebunden** (modal) angezeigt..
- Konstante ***Hinweis*** Der Text dieser Konstante erscheint im oberen Teile des Dialoges als Hinweis für den Benutzer:

VBA-Code

```
'**** Anfang des anpassbaren Teils *****
Private Const Modus = 0 '0 = Ordner wählen; 1: = Datei wählen
Private Const Modal = 1 '0 = ungebunden; 1 = gebunden (modal)
Private Const Hinweis = "Bitte wählen Sie aus:" 'Erscheint wörtlich im Dialog
'**** Ende des anpassbaren Teils *****

Public Type BROWSEINFO
            hOwner As Long
            pidlRoot As Long
            pszDisplayName As String
            lpszTitle As String
            ulFlags As Long
            lpfn As Long
            lParam As Long
            iImage As Long
End Type

Declare Function GetActiveWindow Lib "user32" () As Long
Declare Function SHGetPathFromIDList Lib "shell32.dll" Alias "SHGetPathFromIDListA" _
  (ByVal pidl As Long, ByVal pszPath As String) As Long
Declare Function SHBrowseForFolder Lib "shell32.dll" Alias "SHBrowseForFolderA" _
  (lpBrowseInfo As BROWSEINFO) As Long

Sub VerzeichnisOderDateiAuslesen()
  Antwort = GetDirectory(Hinweis)
  MsgBox Antwort
End Sub

Function GetDirectory(Optional Msg) As String
  Dim bInfo As BROWSEINFO
  Dim path As String
  Dim r As Long, x As Long, pos As Integer
  If Modal = 0 Then
    bInfo.hOwner = 0 'Ungebunden
  Else
    bInfo.hOwner = GetActiveWindow 'Gebunden (modal)
  End If
  bInfo.lpszTitle = Msg 'Titel des Dialoges
  If Modus = 0 Then
    bInfo.ulFlags = &H1 'Ordner auswählen
  Else
    bInfo.ulFlags = &H4000 'Datei auswählen
  End If
  bInfo.pidlRoot = 0& 'Root
  x = SHBrowseForFolder(bInfo)
  path = Space$(512)
  r = SHGetPathFromIDList(ByVal x, ByVal path)
  If r Then
    pos = InStr(path, Chr$(0))
    GetDirectory = Left(path, pos - 1)
  Else
    GetDirectory = ""
  End If
End Function
```

(Windows API, welches entweder einen Ordner bzw. eine Datei zur Auswahl stellt.)

Dialog zur Auswahl eines Ordners oder einer Datei (Windows-API) aus einer UserForm

▲ Zurück zum Anfang

Wenn Sie den Dialog aus einer UserForm aufrufen möchten, dann können Sie einen sehr ähnlichen Code benutzen.

Da einerseits API-Deklarationsanweisungen nicht im Code-Teil einer UserForm stehen dürfen, und da andererseits das UserForm-Fenster für das Betriebssystem unter einer anderen Window-ID läuft, als das Dokumentfenster, müssen Sie den Code aber aufteilen. Was Sie konkret tun sollten, wird nun detailliert geschildert.

Implementierung

- Wählen Sie im VBA-Editor ***Einfügen - Modul***.
- Kopieren Sie den ersten (grösseren) Code in dieses Standardmodul.
- Fügen Sie nun dem Projekt eine neue UserForm bei.
- Plazieren Sie auf diese UserForm eine Befehlsschaltfläche mit dem Namen **CommandButton1**.
- Kopieren Sie den zweiten (kürzeren) Code in den Code-Teile dieser UserForm.
- Soll der Benutzer ein Ordner auswählen, dann lautet der Aufruf wie im nachfolgenden Beispiel: `GetDirectory(GetActiveWindow, 0, Hinweis)` Soll der Benutzer eine Datei auswählen, dann lautet der Aufruf dagegen: `GetDirectory(GetActiveWindow, 1, Hinweis)`

VBA-Code

Code Für das Standardmodul

Kopieren Sie folgenden Code in ein neues Standardmodul des Projekts:

```
Public Type BROWSEINFO
            hOwner As Long
            pidlRoot As Long
            pszDisplayName As String
            lpszTitle As String
            ulFlags As Long
            lpfn As Long
            lParam As Long
            iImage As Long
End Type

Declare Function GetActiveWindow Lib "user32" () As Long
Declare Function SHGetPathFromIDList Lib "shell32.dll" Alias "SHGetPathFromIDListA" _
  (ByVal pidl As Long, ByVal pszPath As String) As Long
Declare Function SHBrowseForFolder Lib "shell32.dll" Alias "SHBrowseForFolderA" _
  (lpBrowseInfo As BROWSEINFO) As Long

Function GetDirectory(Owner As Long, Modus As Long, Optional Msg) As String
  Dim bInfo As BROWSEINFO
  Dim path As String
  Dim r As Long, x As Long, pos As Integer
  bInfo.hOwner = GetActiveWindow 'Gebunden (modal)
  bInfo.lpszTitle = Msg 'Titel des Dialoges
  If Modus = 0 Then
    bInfo.ulFlags = &H1 'Ordner auswählen
  Else
    bInfo.ulFlags = &H4000 'Datei auswählen
  End If
  bInfo.pidlRoot = 0& 'Root
  x = SHBrowseForFolder(bInfo)
  path = Space$(512)
  r = SHGetPathFromIDList(ByVal x, ByVal path)
  If r Then
    pos = InStr(path, Chr$(0))
    GetDirectory = Left(path, pos - 1)
  Else
    GetDirectory = ""
  End If
End Function
```

(Deklarationsanweisungen von Windows-API müssen ausserhalb einer UserForm liegen.)

Code für die UserForm

Aus der UserForm können Sie jetzt das Windows-API, wie folgt aufrufen. Falls der Benutzer eine Datei statt einen Ordner auswählen soll, dann passen Sie den entsprechend Code an, wie dies Eingangs dieses Kapitels beschrieben wurde.

```
Private Sub CommandButton1_Click()
  Hinweis = "Bitte wählen Sie:"
  Antwort = GetDirectory(GetActiveWindow, 0, Hinweis)
  MsgBox Antwort
End Sub
```

(Das Windows-API wird aus der UserForm aufgerufen. Der aufgerufene Dialog ist modaler Natur. Um statt eines Ordners eine Datei zur Auswahl zu stellen, codieren Sie eine *1* statt einer *0*.)

Links zu diesem Thema

▲ Zurück zum Anfang

Microsoft Word MVP FAQ Site

► Calling FileOpen dialog in VBA does not allow opening of multiple files

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Browse.htm

René Probst, Dezember 2000

Änderungen

- --- nil ---

Wie kann ich feststellen, wie oft ein bestimmtest Wort im Dokument vorkommt?

Wie kann ich beim wiederholten Ersetzen einer Zeichenfolge ausweisen, wie viele Ersetzungen getätigt wurden?

Wie kann ich eine Worthäufigkeitsstatistik eines Word-Dokumentes erstellen?

◄ Zurück zum Hauptindex

Diese Seite

- ▼ Ausweisen, wie oft eine vorgegebene Zeichenfolge im Dokumentes auftaucht
- ▼ Ausweisen, wie oft eine vorgegebene Zeichenfolge im Dokument unter Berücksichtigung von Dokumentteilen, wie Kopf-/Fusszeile, Fussnoten etc. auftaucht
- ▼ Ausweisen, wie viele Ersetzungen getätigt wurden
- ▼ Worthäufigkeitsstatistik eines Word-Dokumentes erstellen
- ▼ Links zu diesem Thema

Ausweisen, wie oft eine vorgegebene Zeichenfolge im Hauptteil des Dokumentes auftaucht

▲ Zurück zum Anfang

Ohne Einsatz von VBA

Um ohne VBA die Häufigkeit eines Wortes bzw. einer beliebigen Zeichenfolge im Hauptteil des Dokumentes zu zählen, können Sie wie folgt vorgehen.

1. Speichern Sie das Dokument unter einem anderen Namen. Dabei handelt es sich um eine reine Vorsichtsmassnahme, falls Sie sich in der Folge vertun.
2. Rufen Sie vom Word-Hauptmenü den Dialog ***Bearbeiten - Ersetzen...*** auf.
3. Geben Sie im Feld *Suchen nach:* das Wort bzw. die Zeichenfolge ein, nach welcher gesucht werden soll.
4. Betätigen Sie nun die Schaltfläche mit der Bezeichnung **Alle ersetzen**.
5. Der Assistent zeigt Ihnen nun an, wie viele Ersetzungen vorgenommen wurden.
6. Merken Sie sich die Zahl und drücken Sie unmittelbar danach auf der Symbolleiste mit der Bezeichnung **Standard** die Schaltfläche **Rückgängig**.

Mit Einsatz von VBA

Im folgenden wird eine VBA-Routine aufgezeigt, welche ausweist, wie oft ein Wort bzw. eine Zeichenfolge im Hauptteil des Dokumentes erscheint.

► **So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl**

```
Sub WortZählen()
  Application.ScreenUpdating = False
  Titel = "Wort zählen"
  Antwort = InputBox("Nach welchem Wort soll gesucht werden?", Titel)
  If Antwort = "" Then Exit Sub
  Dim oRange As Range, Anz As Long
  Set oRange = Selection.Range
  ActiveDocument.Range(0, 0).Select
  With Selection.Find
    .ClearFormatting
    .Text = Antwort
    .Execute
    While .Found = True
      Anz = Anz + 1
      .Execute
    Wend
  End With
  oRange.Select
  Application.ScreenUpdating = True
  Q = Chr(34)
  strT = "Für das Wort " & Q & Antwort & Q & " wurden " & _
     Anz & " Entsprechungen gefunden."
  MsgBox strT, vbInformation, Titel
End Sub
```

(Ermittelt wie oft ein Wort bzw. eine Zeichenfolge im Hauptteil des Dokumentes vorkommt.)

Alle Dokumententeile berücksichtigen

▲ Zurück zum Anfang

Die folgende Routine ist vergleichbar mit der vorhergehenden. Diesmal wird aber nicht nur der Hauptteil des Dokumentes berücksichtigt sondern auch die übrigen Dokumentteile namentlich:

- **Hauptteil** des Dokumentes
- **Primäre Kopfzeile** (falls vorhanden)
- **Erste Seite Kopfzeile** (falls vorhanden)
- **Gerade Seite Kopfzeile** (falls vorhanden)
- **Primäre Fusszeile** (falls vorhanden)
- **Erste Seite Fusszeile** (falls vorhanden)
- **Gerade Seite Fusszeile** (falls vorhanden)
- Sämtliche **Textfelder** (falls vorhanden)
- Der **Kommentar**bereich (falls vorhanden)
- Sämtliche **Fussnoten**bereiche (falls vorhanden)
- Der **Endnoten**bereich (falls vorhanden)

```
Sub WortZählenInAlleDokumententeilen()
  Application.ScreenUpdating = False
  Titel = "Wörter zählen"
  strIn = InputBox("Nach welchem Wort soll gesucht werden?", Titel)
  If strIn = "" Then Exit Sub
  Dim oStory As Range
  Anz = 0
  For Each oStory In ActiveDocument.StoryRanges
    Anz = Anz + WortSuchen(oStory, strIn)
    While Not (oStory.NextStoryRange Is Nothing)
      Set oStory = oStory.NextStoryRange
      Anz = Anz + WortSuchen(oStory, strIn)
    Wend
  Next
  Application.ScreenUpdating = True
  Q = Chr(34)
  strT = "Für das Wort " & Q & strIn & Q & " wurden " & _
     Anz & " Entsprechungen gefunden."
  MsgBox strT, vbInformation, Titel
End Sub

Private Function WortSuchen(oStory, ByVal Wort As String) As Long
  Flag = False
  Anzahl = 0
  Select Case oStory.StoryType
    Case wdMainTextStory: Flag = True
    Case wdFootnotesStory: Flag = True
    Case wdEndnotesStory: Flag = True
    Case wdCommentsStory: Flag = True
    Case wdTextFrameStory: Flag = True
    Case wdPrimaryHeaderStory: Flag = True
    Case wdFirstPageHeaderStory: Flag = True
    Case wdEvenPagesHeaderStory: Flag = True
    Case wdPrimaryFooterStory: Flag = True
    Case wdFirstPageFooterStory: Flag = True
    Case wdEvenPagesFooterStory: Flag = True
  End Select
  If Flag = False Then Exit Function
  With oStory.Find
    .ClearFormatting
    .Text = Wort
    .Execute
    While .Found = True
      Anzahl = Anzahl + 1
      .Execute
    Wend
  End With
  WortSuchen = Anzahl
End Function
```

(Gleiche Aufgabenstellung, wie im vorhergehenden Beispiel. Diesmal werden aber alle Dokumententeile berücksichtig..)

☛ Bei Bedarf können Sie den Code auf sehr einfache Art und Weise dahin gehend anpassen, dass nur von Ihnen bestimmte Dokumententeile bei der Zählung berücksichtigt werden.

☹ Wegen eines bei Microsoft dokumentierten Fehlerverhaltens von MS Word, werden Textfelder innerhalb von Kopf- oder Fusszeilen nicht berücksichtigt. Siehe dazu den Link am Ende dieses Skriptes.

Ausweisen, wie viele Ersetzungen getätigt wurden

▲ Zurück zum Anfang

Wenn Sie von Wort-Hauptmenü aus den Dialog ***Bearbeiten - Ersetzen...*** einsetzen, wird Ihnen nach dem Ersetzungsvorgang entweder in der Statuszeile oder durch den Office-Assistenten avisiert, wie viele Ersetzungen getätigt wurden.

Leider ist es nicht möglich, diese Information ex Word-VBA anzuzapfen. Möchten Sie also diesen Vorgang ex VBA nachvollziehen und gleichzeitig diese Information dem Benutzer transparent machen, dann müssen Sie ein etwas plumpes und unperformandes Vorgehen wählen.

Den VBA-Code, welchen Sie dazu benötigen, wird gleich anschliessend vorgestellt. Beachten Sie, dass bei dieser Version, ausschliesslich die *Hits* im Hauptteil des Dokumentes ersetzt und ausgewiesen werden.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub WortErsetzenUndZaehlen()
  Application.ScreenUpdating = False
  Titel = "Wort ersetzen"
  strIn = InputBox("Nach welchem Wort soll gesucht werden?", Titel)
  If strIn = "" Then Exit Sub
  strAus = InputBox("Mit welchem Wort soll das fraglich Wort ersetzt werden?", Titel)
  If strAus = "" Then Exit Sub
  DoEvents
  Anz = 0
  Dim oRange As Range
  Set oRange = Selection.Range
  ActiveDocument.Range(0, 0).Select
  With Selection.Find
    .ClearFormatting
    .Text = strIn
    .Replacement.Text = strAus
    .Execute Replace:=wdReplaceOne
    While .Found = True
      Selection.Collapse Direction:=wdCollapseEnd
      Anz = Anz + 1
      .Execute Replace:=wdReplaceOne
    Wend
  End With
  oRange.Select
  Application.ScreenUpdating = True
  strT = "Es wurden " & Anz & " Ersetzungen getätigt."
  MsgBox strT, vbInformation, Titel
End Sub
```

(Erlaubt es ein Wort bzw. eine Zeichenfolge mit einer anderen Zeichenfolge zu ersetzen. Am Schluss wird ausgewiesen, wie viele Ersetzungen getätigt wurden. Berücksichtigt wird ausschliesslich der Hauptteil des Dokumentes.)

Alle Dokumententeile berücksichtigen

In Anlehnung zum vorgängigen Kapitel erfolgt nun noch die Lösung, welche alle Dokumententeile bei der Ersetzung und bei der Zählung berücksichtigt.

```
Sub WortInAllenDokumententeilenErsetzenUndZaehlen()
  Application.ScreenUpdating = False
  Titel = "Wort ersetzen"
  strIn = InputBox("Nach welchem Wort soll gesucht werden?", Titel)
  If strIn = "" Then Exit Sub
  strAus = InputBox("Mit welchem Wort soll das fraglich Wort ersetzt werden?", Titel)
  If strAus = "" Then Exit Sub
  DoEvents
  Anz = 0
  Dim oStory As Range
  For Each oStory In ActiveDocument.StoryRanges
    Anz = Anz + WortErsetzen(oStory, strIn, strAus)
    While Not (oStory.NextStoryRange Is Nothing)
      Set oStory = oStory.NextStoryRange
      Anz = Anz + WortErsetzen(oStory, strIn, strAus)
    Wend
  Next
  Application.ScreenUpdating = True
  strT = "Es wurden " & Anz & " Ersetzungen getätigt."
  MsgBox strT, vbInformation, Titel
End Sub

Private Function WortErsetzen(oStory As Range, ByVal Wort1 As String, ByVal Wort2 As String)
  Anzahl = 0
  Flag = False
  Select Case oStory.StoryType
    Case wdMainTextStory: Flag = True
    Case wdFootnotesStory: Flag = True
    Case wdEndnotesStory: Flag = True
    Case wdCommentsStory: Flag = True
    Case wdTextFrameStory: Flag = True
    Case wdPrimaryHeaderStory: Flag = True
    Case wdFirstPageHeaderStory: Flag = True
    Case wdEvenPagesHeaderStory: Flag = True
    Case wdPrimaryFooterStory: Flag = True
    Case wdFirstPageFooterStory: Flag = True
    Case wdEvenPagesFooterStory: Flag = True
  End Select
  If Flag = False Then Exit Function
  With oStory.Find
    .ClearFormatting
    .Text = Wort1
    .Replacement.Text = Wort2
    .Execute Replace:=wdReplaceOne
    While .Found = True
      Anzahl = Anzahl + 1
      .Execute Replace:=wdReplaceOne
    Wend
  End With
  WortErsetzen = Anzahl
End Function
```

(Ersetzt in allen Dokumententeile eine Zeichenfolge mit einer anderen. Am Ende wird ausgewiesen, wie viele Ersetzungen getätigt wurden.)

Bei Bedarf können Sie den Code auf sehr einfache Art und Weise dahin gehend anpassen, dass nur von Ihnen bestimmte Dokumententeile bei der Ersetzung berücksichtigt werden.

Wegen eines bei Microsoft dokumentierten Fehlerverhaltens von MS Word, werden Textfelder innerhalb von Kopf- oder Fusszeilen nicht berücksichtigt. Siehe dazu den Link am Ende dieses Skriptes.

Worthäufigkeitsstatistik eines Word-Dokumentes erstellen

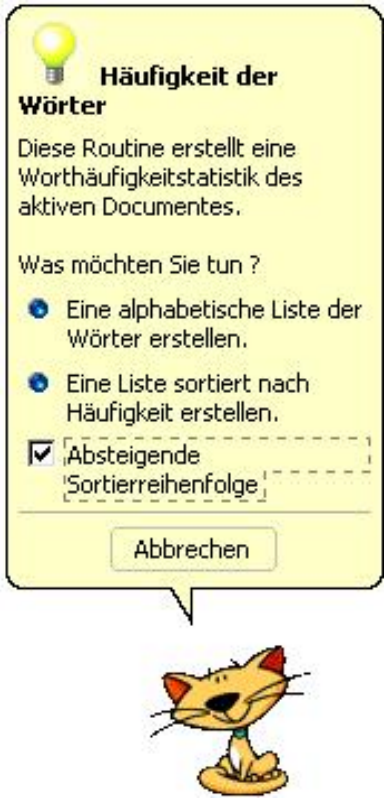
▲ Zurück zum Anfang

Im folgenden wir eine sehr komfortable und effiziente Lösung vorgestellt, welche eine Häufigkeitsstatistik aller Wörter im aktiven Dokument erstellt. Das Resultat wird dann in einer Tabelle präsentiert. Sie können beeinflussen, ob die Statistik die Wörter alphabetisch aufzeigen soll, oder aber nach Häufigkeit. Berücksichtigt wird der Text im Hauptteil des Dokumentes.

Kopieren Sie den nachfolgenden VBA-Code in ein neues Modul (z.B. der Standardvorlage mit dem Namen *Normal.dot)* und testen Sie das Programm.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
'
' Diese Word-Routine erstellt eine Worthäufigkeitsstatistik' des aktiven Dokumentes.
' Erstellet am 08/03/99 von René Probst / eku / CH-8058-Glattbrugg
'
Option Compare Text
Private asTitle As String
Private asVisible As Boolean
Private oDoc As Document
Private sDoc As Document
Private wDoc As Document
Private oBlase As Balloon
Private Wahl As Integer
Private SortFeld As Integer
Private SortOrder As Integer
Private strWork As String
Private Stat()
Private TotWort As Double
Private AnzWort As Integer
Private LangWort As Integer
Private TotLänge As Double
Private Zähler As Integer

Sub Worthäufigkeit()
  LangWort = 0
  TotLänge = 0
  TotWort = 0
  asVisible = Assistant.Visible
  asTitle = "Häufigkeit der Wörter"
  Set oDoc = ActiveDocument
  If oDoc.Content = Chr(13) Then WortKeineWörter
  Assistant.Visible = True
  Assistant.Animation = msoAnimationAppear
  Wahl = BlaseAnzeigen
  Select Case Wahl
```



```
    Case 1, 2
      SortFeld = Wahl
      If oBlase.Checkboxes(1).Checked = True Then
        SortOrder = wdSortOrderDescending
      Else
        SortOrder = wdSortOrderAscending
      End If
    Case -2
      HappyEnd
  End Select
  If oDoc.Words.Count > 8191 Then
    ReDim Stat(8191, 1)
  Else
    ReDim Stat(oDoc.Words.Count, 1)
  End If
  Zähler = -1
  Assistant.Animation = msoAnimationWritingNotingSomething
  For Each oWort In ActiveDocument.Words
    If Left(UCase(oWort), 1) Like "[A-ZÄÖÜÀÂÉÈÊÎÏÔÛ]" Then WortAufnehmen oWort
  Next
  strWork = ""
  AnzWort = Zähler + 1
  If AnzWort > 0 Then
    WortAuflisten
    WortTabelle
    WortDokEinrichten
    WortZusammenfassung
  Else
    WortKeineWörter
  End If
  HappyEnd
End Sub

Private Sub WortAufnehmen(tWort)
  TotWort = TotWort + 1
  tWort = RTrim(tWort)
  ofs = InStr(tWort, "-")
  While ofs > 0
    tWort = Left(tWort, ofs - 1) + Right(tWort, Len(tWort) - ofs)
    ofs = InStr(tWort, "-")
  Wend
  TotLänge = TotLänge + Len(tWort)
  If Len(tWort) > LangWort Then LangWort = Len(tWort)
  chk = InStr(strWork, Chr(1) + tWort + Chr(2))
  If chk = 0 Then
    StatusBar = CStr(TotWort) + " Wörter verarbeitet - " + CStr(Zähler + 1) + " _
       Wörter aufgenommen...bitte warten."
    DoEvents
    Zähler = Zähler + 1
    strWork = strWork + Chr(1) + tWort + Chr(2) + Format(Zähler, "0000")
    Stat(Zähler, 0) = tWort
    Stat(Zähler, 1) = 1
  Else
```

```
    ofs = InStr(chk, strWork, Chr(2))
    j = Mid(strWork, ofs + 1, 4)
    Stat(j, 1) = Stat(j, 1) + 1
  End If
End Sub

Private Sub WortAuflisten()
  Documents.Add
  ActiveWindow.View = wdNormalView
  Assistant.Animation = msoAnimationWorkingAtSomething
  StatusBar = "Die Liste der Wörter wird erstellt."
  DoEvents
  Set wDoc = ActiveDocument
  Application.ScreenUpdating = False
  Selection.TypeText Text:="Wort" + vbTab + "Hits" + vbCrLf
  For j = 0 To AnzWort - 1
    Selection.TypeText Text:=Stat(j, 0) + vbTab + CStr(Stat(j, 1)) + vbCrLf
  Next j
  ReDim Stat(0, 0)
End Sub

Private Sub WortTabelle()
  Assistant.Animation = msoAnimationGetTechy
  StatusBar = "Die Liste der Wörter wird sortiertt."
  DoEvents
  With Selection
    .WholeStory
    .ConvertToTable Separator:=wdSeparateByTabs, Format:=wdTableFormatProfessional, _
       AutoFit:=True
    Dim oTable As Table
    Set oTable = wDoc.Tables(1)
    If AnzWort > 1 Then
      Select Case SortFeld
        Case 1
          oTable.Sort ExcludeHeader:=True, _
            FieldNumber:="Spalte1", SortFieldType:=0, SortOrder:=SortOrder
        Case 2
          oTable.Sort ExcludeHeader:=True, _
            FieldNumber:="Spalte2", SortFieldType:=1, SortOrder:=SortOrder, _
            FieldNumber2:="Spalte1", SortFieldType2:=0, SortOrder2:=0
      End Select
    End If
    oTable.Rows(1).HeadingFormat = True
    oTable.Columns(2).Select
    Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
    Selection.Cells(1).Select
    Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
    oTable.Select
  End With
End Sub

Private Sub WortDokEinrichten()
  Assistant.Animation = msoAnimationGetArtsy
```

```
  Documents.Add
  ActiveWindow.View = wdNormalView
  StatusBar = "Die Statistik wird erstellt."
  DoEvents
  Set sDoc = ActiveDocument
  Selection.Style = "Überschrift 1"
  Selection.TypeText Text:="Häufigkeit der Wörter in " + oDoc.Name
  Selection.TypeParagraph
  Set sKZ = sDoc.Sections(1).Headers(1).Range
  sDoc.Fields.Add Range:=sKZ, Type:=wdFieldEmpty, Text:="FVREF ""Überschrift 1"""
  sDoc.Sections(1).Footers(1).Range.Select
  With Selection
    .InsertAfter "Seite "
    .EndKey
    .Fields.Add Range:=Selection.Range, Type:=wdFieldPage
    .EndKey
    .InsertAfter " von "
    .EndKey
    .Fields.Add Range:=Selection.Range, Type:=wdFieldEmpty, Text:="ANZSEITEN "
    .ParagraphFormat.Alignment = wdAlignParagraphRight
  End With
  ActiveWindow.ActivePane.Close
End Sub

Private Sub WortZusammenfassung()
  Häufigkeit = Format(TotWort / AnzWort, "0")
  Durchschnitt = Format(TotLänge / TotWort, "0")
  With Selection
    .TypeText Text:="Anzahl Wörter:" + vbTab + CStr(TotWort) + Chr(11)
    .TypeText Text:="Anzahl Einträge:" + vbTab + CStr(AnzWort) + Chr(11)
    .TypeText Text:="Durchschnittliche Häufigkeit:" + vbTab + CStr(Häufigkeit) + _
       Chr(11)
    .TypeText Text:="Längstes Wort:" + vbTab + CStr(LangWort) + Chr(11)
    .TypeText Text:="Durchschnittliche Wortlänge:" + vbTab + Durchschnitt
    With .ParagraphFormat
      .Style = "Standard"
      .TabStops.Add CentimetersToPoints(6), Alignment:=wdAlignTabRight
      .SpaceBefore = 6
      .SpaceAfter = 18
    End With
    .TypeParagraph
    .Range.FormattedText = wDoc.Content
  End With
End Sub

Private Sub HappyEnd()
  If Wahl > 0 And AnzWort > 0 Then
    wDoc.Close SaveChanges:=wdDoNotSaveChanges
    sDoc.Activate
    Selection.HomeKey Unit:=wdStory
  End If
  Assistant.Visible = asVisible
  StatusBar = ""
```

```
  Application.ScreenUpdating = True
  End
End Sub

Private Sub WortKeineWörter()
  Antwort = MsgBox("Dieses Dokument enthält keine Wörter.", vbCritical, asTitle)
  HappyEnd
End Sub

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = asTitle
    asText = "Diese Routine erstellt eine Worthäufigkeitstatistik des aktiven Docs."
    .Text = asText + vbCrLf + vbCrLf + "Was möchten Sie tun ?"
    .Labels(1).Text = "Eine alphabetische Liste der Wörter erstellen."
    .Labels(2).Text = "Eine Liste sortiert nach Häufigkeit erstellen."
    .Checkboxes(1).Text = "Absteigende Sortierreihenfolge"
    .Button = msoButtonSetCancel
    BlaseAnzeigen = .Show
  End With
End Function
```

(Erstellt in einem neuen Dokument eine Worthäufigkeitsstatistik des aktiven Dokumentes.)

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Der Befehl Suchen/Ersetzen funktioniert nicht in Kopf- und Fusszeilen und auch nicht in Textfeldern

Microsoft

► WD2000: StoryRanges Does Not Return Text Box Text in Header/Footer

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Worth.htm

René Probst, Januar 2001

Änderungen

- --- nil ---

Wie kann ich die Verzeichnisstruktur einer Festplattenpartition, CD-ROM, Diskette oder eines Hauptverzeichnisses in einem Word-Dokument ausgeben?

◄ Zurück zum Hauptindex

Problem

> Ich bin auf der Suche nach einer Routine, welche mir die Verzeichnisstruktur meiner CD-ROM übersichtlich in einem Word-Dokument darstellt und mir auch den Ausdruck ermöglicht.

Lösung

Im folgenden wird eine VBA-Routine vorgestellt, welche es Ihnen ermöglicht, die Verzeichnisstruktur einer Festplattenpartition, einer CD-ROM, einer Diskette oder allenfalls auch eines von Ihnen bestimmten Hauptverzeichnisses übersichtlich in einem Word-Dokument auszugeben. Auf Wunsch kann das Ergebnis dann gedruckt werden. Dieser Routine arbeitet mit dem Office-Assistenten zusammen und ist sehr einfach zu bedienen.

Funktionalität

Im Folgenden erfahren Sie, was die vorgeschlagene Routine kann, bzw. nicht kann.

- Die Routine schlägt Ihnen die Laufwerke (bzw. Festplattenpartitionen), welche auf dem Rechner vorhanden sind, zur Auswahl vor.
- Statt eines Laufwerkes können Sie auch ein beliebiges Hauptverzeichnis, welches auf dem Rechner existiert, vorgeben.
- Die Verzeichnisse werden hierarchisch und alphabetisch korrekt sortiert aufgelistet.
- Die Hierarchie eines Unterverzeichnis wird durch Einrücken und Grösse der Schrift in dem der Namen des Verzeichnisses erscheint, visualisiert.
- Auf Wunsch können Sie sich auch anzeigen lassen wie viele Dateien sich jeweils in einem aufgelisteten Verzeichnis befinden.
- Weiter auf Wunsch können Sie sich auch anzeigen lassen wie viel Platz jedes der aufgelisteten Verzeichnisse beansprucht.
- Die Routine listet ausschliesslich Verzeichnisse auf, nicht aber einzelne Dateien.

! Bitte beachten Sie, dass die Routine das CD-ROM-Laufwerk nur dann zur Auswahl stellt, wenn dieses mit einer gültigen CD-ROM geladen und bereit ist. Analoges gilt für das Diskettenlaufwerk.

☛ Sie finden am Ende dieses Skripts einen Link zu einer Software mit dem Namen **Directory Printer 3.1**, welche als Shareware vertrieben wird und u.a. nicht nur die Struktur der Verzeichnisse auflisten und drucken kann, sondern auch die einzelnen Dateien, welche sich in diesen Verzeichnissen befinden. Ich habe diese Software seit langem im Einsatz und kann sie weiter empfehlen.

Lösung

So bringen Sie die Lösung zum Laufen

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** über den Word-Dialog ***Datei - Öffnen***.
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
3. Dort angekommen, wählen Sie vom Hauptmenü ***Einfügen - Modul***.
4. Kopieren Sie nun den unten abgedruckten VBA-Code in das leere Fenster, welches sich geöffnet hat.
5. Speichern Sie nun die Dokumentvorlage und schliessen Sie diese.
6. Rufen Sie nun über ***Extras - Makro - Makros...*** das Makro mit dem Namen **Verzeichnisbaum** auf.
7. Zusätzliche Informationen, wie Sie ein Makro in Ihrer Word-Umgebung komfortabel einbetten können, erfahren Sie im folgenden Skript.

► **So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl.**

VBA-Code

```
' Diese Routine liest Verzeichnisse und Unterverzeichnisse eines von
' Ihnen gewählten Laufwerkes aus und stellt diese in einer Baumstruktur dar.
'
' Erstellt am 15/05/99 von René Probst / eku / CH-8152 Glattbrugg
'
' Bitte beachten Sie, dass das CD-ROM-Laufwerk nur dann zur Auswahl
' steht, wenn ein gültiger Datenträger eingelegt ist.
'
Private Const Steps = 0.5
Private Const Schrift = 13
Private Const F1 = "Der Datenträger ist nicht bereit oder ein Fehler ist aufgetreten."
Private Const F2 = "Zusatzinformation: Fehler-Code: "
Private Const F3 = "Fehlerbeschreibung: "
Private Const I1 = ": *nicht bereit*"
Private Const I2 = "Anderes"
Private Const I3 = "Anderes (Benutzerdefiniert)"
Private Const I4 = "Geben Sie ein Laufwerk oder ein Verzeichnis an für den Verzeichnisbaum."
Private Const I5 = "Der von Ihnen eingegebene Pfad "
Private Const I6 = " ist ungültig oder existiert nicht."
Private Const asTitle = "Verzeichnisbaum darstellen"
Private Const Titel = "Verzeichnisbaum des Laufwerkes"
Private Const Füller = 0 '0 = Keine; 1 = Punkte; 2 = Striche; 3 = Linie
Private asVisible As Boolean
Private Pfad As String
Private Verz() As String
Private sVerz() As String
Private LW()
Private LWBez()
Private MaxLW As Integer
Private tmp() As String
```

```
Private tmpstr As String
Private Z As Integer
Private AnzDatei As Double
Private TotSize As Double
Private Tabs() As Double
Private Marke As Variant
Private Wahl As String
Private oBlase As Balloon
Private blPoint As Integer
Private blLabel() As String
Private blButton As Integer
Public rDoc As Document

Sub Verzeichnisbaum()
  Set Test = Application.Assistant
  If Test Is Nothing Then
    strT = "Der Office-Assistent ist nicht oder nicht richtig installiert."
    MsgBox strT, vbCritical, asTitle
    Exit Sub
  End If
  Set Test = Nothing
  asVisible = Assistant.Visible
  VerzeichnisBaumLaufwerke
  VerzeichnisBaumAbfrage
  If Not Wahl = "0" Then
    VerzeichnisBaumEinlesen
    VerzeichnisBaumSortieren
    VerzeichnisBaumSeiteEinrichten
    VerzeichnisBaumDarstellen
  End If
  HappyEnd
End Sub

Private Sub VerzeichnisBaumLaufwerke()
  j = 0
  For i = 65 To 90
    On Error Resume Next
    ChDrive Chr(i)
    Fehler = Err.Number
    On Error GoTo 0
    If Fehler = 0 Or i = 65 Then
      j = j + 1
      ReDim Preserve LW(j)
      ReDim Preserve LWBez(j)
      LW(j) = Chr(i) + ":"
      If Fehler = 0 Then
        lab = Dir(Chr(i) + ":\", vbVolume)
        If lab = "" Then
          LWBez(j) = Chr(i) + ":"
        Else
          LWBez(j) = Chr(i) + ": (" + UCase(Left(lab, 1)) + _
             LCase(Right(lab, Len(lab) - 1)) + ")"
        End If
```

```
      End If
      If i = 65 And Fehler > 0 Then LWBez(j) = Chr(i) + I1
    End If
  Next i
  j = j + 1
  ReDim Preserve LW(j)
  ReDim Preserve LWBez(j)
  LW(j) = I2
  LWBez(j) = I3
  MaxLW = j
End Sub

Private Sub VerzeichnisBaumAbfrage()
  Assistant.Visible = True
  Assistant.Animation = msoAnimationBeginSpeaking
  DoEvents
  blPoint = 1
  AssignLabels
  Wahl = BlaseAnzeigen
  If Wahl = "0" Then Exit Sub
  If Wahl = MaxLW Then
    Flag = False
    While Flag = False
      Antwort = InputBox(I4, asTitle)
      If Antwort = "" Then
        Verzeichnisbaum
        Exit Sub
      End If
      On Error Resume Next
      ChDir (Antwort)
      Fehler = Err.Number
      On Error GoTo 0
      If Fehler > 0 Then
        Q = Chr(34)
        Dummy = MsgBox(I5 + Q + Antwort + Q + I6, vbCritical, asTitle)
      Else
        Flag = True
        Pfad = Antwort + "\"
      End If
    Wend
  Else
    Pfad = LW(Wahl) + "\"
  End If
End Sub

Private Sub VerzeichnisBaumEinlesen()
  ReDim Verz(8191, 3)
  tmpstr = Chr(1) + Pfad + Chr(2) + "0001"
  Verz(1, 0) = Pfad
  Verz(1, 1) = Pfad
  Z = 1
  pos = 1
  While Verz(pos, 1) <> ""
```

```
    Reporting 0, Left(Verz(pos, 1), Len(Verz(pos, 1)) - 1)
    DirRead Verz(pos, 1)
    Verz(pos, 2) = AnzDatei
    Verz(pos, 3) = TotSize
    pos = pos + 1
  Wend
End Sub

Private Sub VerzeichnisBaumSortieren()
  Reporting 1, "Die Verzeichnisse werden sortiert."
  ReDim sVerz(Z, 5)
  For i = 1 To Z
    idx = ""
    ofs = 1
    ofs = InStr(ofs, Verz(i, 1), "\")
    Ebene = 0
    While ofs > 0
      Ebene = Ebene + 1
      chk = Left(Verz(i, 1), ofs)
      pos = InStr(Chr(1) + tmpstr + Chr(2), chk)
      idx = idx + Mid(tmpstr, InStr(pos, tmpstr, Chr(2)) + 1, 4)
      ofs = InStr(ofs + 1, Verz(i, 1), "\")
    Wend
    For j = Len(idx) To 36 Step 4
      idx = idx + "0000"
    Next j
    sVerz(i, 0) = idx
    sVerz(i, 1) = UCase(Left(Verz(i, 0), 1)) + _
       LCase(Right(Verz(i, 0), Len(Verz(i, 0)) - 1))
    sVerz(i, 2) = Verz(i, 1)
    sVerz(i, 3) = Ebene
    sVerz(i, 4) = Verz(i, 2)
    sVerz(i, 5) = Verz(i, 3)
  Next i
  WordBasic.SortArray sVerz()
  ReDim Verz(0, 0)
  tmpstr = ""
End Sub

Private Sub VerzeichnisBaumDarstellen()
  Reporting 1, "Die Liste der Verzeichnisse wird erstellt."
  Application.ScreenUpdating = False
  With Selection
    .Style = wdStyleHeading1
    .TypeText Text:=Titel + " (" + Left(Pfad, Len(Pfad) - 1) + ")"
    .ParagraphFormat.SpaceAfter = 9
    .TypeParagraph
    .LanguageID = wdNoProofing
    .Style = wdStyleNormal
    With .ParagraphFormat
      .TabStops.ClearAll
      i = 1
      While Tabs(i) <> 0
```

```
        .TabStops.Add Position:=Tabs(i), Alignment:=wdAlignTabRight, Leader:=Füller
        i = i + 1
      Wend
    End With
  End With
  For i = 2 To Z
    With Selection
      .ParagraphFormat.LeftIndent = CentimetersToPoints(Steps * (sVerz(i, 3) - 2))
      .Font.Size = Schrift - sVerz(i, 3) + 2
      .TypeText Text:=sVerz(i, 1)
      If oBlase.Checkboxes(1).Checked = True Then .TypeText Text:=vbTab + sVerz(i, 4)
      If oBlase.Checkboxes(2).Checked = True Then .TypeText Text:=vbTab + _
         Grösse(sVerz(i, 5))
      .TypeParagraph
    End With
  Next i
  Selection.Style = wdStyleNormal
  If Z = 1 Then
    Selection.Font.Italic = True
    Selection.TypeText Text:="Keine"
    Selection.TypeParagraph
    Selection.Font.Italic = False
  End If
End Sub

Private Sub VerzeichnisBaumSeiteEinrichten()
  On Error Resume Next
  rDoc.Activate
  If Err.Number > 0 Then
    Set rDoc = Documents.Add
    rDoc.Activate
    FusszeileEinrichten
  Else
    On Error GoTo 0
    Selection.EndKey unit:=wdStory
    Selection.InsertBreak Type:=wdPageBreak
    Selection.EndKey unit:=wdStory
  End If
  On Error GoTo 0
  Marke = "M" & Format(Time, "hhmmss")
  Selection.Bookmarks.Add Name:=Marke
  Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
  pos = Selection.Information(wdHorizontalPositionRelativeToPage)
  Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
  ReDim Tabs(3)
  Select Case oBlase.Checkboxes(1).Checked + oBlase.Checkboxes(2).Checked
    Case -1
      Tabs(1) = pos
    Case -2
      Tabs(1) = pos
      Tabs(2) = pos - 80
  End Select
End Sub
```

```
Private Sub FusszeileEinrichten()
  rDoc.Sections(1).Footers(1).Range.Select
  With Selection
    .Fields.Add Range:=Selection.Range, Type:=wdFieldEmpty, _
       Text:="FVREF ""Überschrift 1"""
    .EndKey
    .TypeText Text:=vbTab
    .InsertAfter "Seite "
    .EndKey
    .Fields.Add Range:=Selection.Range, Type:=wdFieldPage
    .EndKey
    .InsertAfter " von "
    .EndKey
    .Fields.Add Range:=Selection.Range, Type:=wdFieldEmpty, Text:="ANZSEITEN "
  End With
  With Selection.ParagraphFormat
    .TabStops.ClearAll
    .Alignment = wdAlignParagraphRight
     pos = Selection.Information(wdHorizontalPositionRelativeToPage)
    .Alignment = wdAlignParagraphLeft
    .TabStops.Add Position:=pos, Alignment:=wdAlignTabRight
  End With
  ActiveWindow.ActivePane.Close
End Sub

Private Sub DirRead(tPfad)
  On Error Resume Next
  Flag = False
  While Flag = False
    chk = Dir(tPfad, 22)
    If Err.Number > 0 Then
      temp = vbCrLf + vbCrLf
      Antwort = MsgBox(F1 + temp + F2 + CStr(Err.Number) + vbCrLf + F3 + _
         Err.Description, 21, asTitle)
      If Antwort = 2 Then
        Wahl = "0"
        Exit Sub
      End If
      Err.Clear
    Else
      Flag = True
    End If
  Wend
  On Error GoTo 0
  i = 0
  AnzDatei = 0
  TotSize = 0
  While chk <> ""
    If (GetAttr(tPfad & chk) And vbDirectory) = vbDirectory Then
      If chk <> "." And chk <> ".." Then
        Z = Z + 1
        i = i + 1
```

```
        ReDim Preserve tmp(i)
        Verz(Z, 0) = chk
        Verz(Z, 1) = tPfad + chk + "\"
        tmp(i) = tPfad + chk + "\"
      End If
    Else
      AnzDatei = AnzDatei + 1
      If oBlase.Checkboxes(2).Checked = True Then
        TotSize = TotSize + FileLen(tPfad & chk)
      End If
    End If
    chk = Dir
  Wend
  If i > 1 Then WordBasic.SortArray tmp()
  Maxtmp = i
  For i = 1 To Maxtmp
    tmpstr = tmpstr + Chr(1) + tmp(i) + Chr(2) + Format(i, "0000")
  Next i
End Sub

Private Sub AssignLabels()
  If MaxLW < 6 Then
    blButton = msoButtonSetCancel
  Else
    Select Case blPoint
      Case 1
        blButton = msoButtonSetNextClose
      Case Else
        If MaxLW < blPoint + 5 Then
          blButton = msoButtonSetBackClose
        Else
          blButton = msoButtonSetBackNextClose
        End If
    End Select
  End If
  ReDim blLabel(6)
  j = 0
  For i = blPoint To blPoint + 4
    j = j + 1
    If UBound(LWBez) >= i Then blLabel(j) = LWBez(i)
  Next i
End Sub

Private Sub Reporting(tType, tMeldung)
  Select Case tType
    Case 0
      rText = "Verzeichnis " + tMeldung + " wird verarbeitet."
    Case 1
      rText = tMeldung
  End Select
  StatusBar = rText
  DoEvents
End Sub
```

```
Private Sub HappyEnd()
  On Error Resume Next
  If Marke <> "" Then
    If rDoc.Bookmarks.Exists(Marke) = True Then
      Selection.GoTo What:=wdGoToBookmark, Name:=Marke
      rDoc.Bookmarks(Marke).Delete
    End If
  End If
  On Error GoTo 0
  Assistant.Visible = asVisible
  Application.ScreenUpdating = True
  StatusBar = ""
End Sub

Private Function Grösse(tGrösse)
  Gr = Format(CDbl(tGrösse) / 1024, "0")
  If tGrösse > 0 And Gr = 0 Then
    Gr = 1
  Else
    Max = Len(Gr) \ 3
    If Max > 0 Then
      ReDim j(Max)
      For i = Max To 1 Step -1
        j(i) = Right(Gr, 3)
        Gr = Left(Gr, Len(Gr) - 3)
      Next i
      j(0) = Gr
      Gr = ""
      For i = 0 To Max
        If j(i) <> "" Then Gr = Gr & "'" & j(i)
      Next i
        Gr = Right(Gr, Len(Gr) - 1)
    End If
  End If
  Grösse = Gr & "kb"
End Function

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = asTitle
   asText = "Diese Routine erstellt einen Verzeichnisbaum eines Laufwerkes Ihrer Wahl."
    .Text = asText + vbCrLf + vbCrLf + "Bitte wählen Sie ein Laufwerk aus:"
    .Checkboxes(1).Text = "Anzahl der Dateien pro Verzeichnis darstellen."
    .Checkboxes(2).Text = "Gesammtgrösse der Dateien pro Verzeichnis darstellen."
    Flag = False
    While Flag = False
      i = 1
      While blLabel(i) <> ""
        .Labels(i).Text = blLabel(i)
```

```
        i = i + 1
      Wend
      .Labels.Count = i - 1
      .Button = blButton
      Wahl = .Show
      If Assistant.BalloonError = True Then Flag = True
      Select Case Wahl
        Case 1 To 5
          Flag = True
          BlaseAnzeigen = Wahl + blPoint - 1
        Case -5
          blPoint = blPoint - 5
          AssignLabels
        Case -6
          blPoint = blPoint + 5
          AssignLabels
        Case -2, -12
          BlaseAnzeigen = 0
          Flag = True
      End Select
    Wend
  End With
End Function
```

(Gibt den Verzeichnisbaum einer Festplattenpartition oder eines vorgegebenen Hauptverzeichnis in einem Word-Dokument aus..)

Links zu diesem Thema

Glenn Alcott

► **Directory Printer 3.1**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/VzBaum.htm

René, Probst, Februar 2001

Änderungen

- --- nil ---

Symbolleistenschaltflächen und Menüleisten selbst gemacht.

◄ Zurück zum Hauptindex

Diese Seite

▼ Ausgangslage
▼ Arten von Schaltflächen
▼ Integrierter Word-Befehl mit einem einfachen Schaltknopf verknüpfen
▼ Bestehendes Makro mit einem einfachen Schaltknopf verknüpfen
▼ Erscheinungsbild eines Schaltknopfes beeinflussen
▼ Schaltknopf mittels VBA erstellen
▼ Ein-/Ausschalter (Toggle-Button)
▼ Wichtige Eigenschaften und Methoden für Schaltknöpfe
▼ Kombinationsfelder (ComboBoxes)
▼ Wichtige Eigenschaften und Methoden für Kombinationsfelder
▼ Neues Menü über Word-Dialog erstellen
▼ Neues Menü mit VBA erstellen
▼ Beispiel für ein Menü: Zugriff auf Textmarken in Word übersichtlicher organisieren
▼ Wichtige Eigenschaften und Methoden für Menüs
▼ Die FindControl-Methode
▼ Die ActionControl-Methode
▼ Der CustomizationContext
▼ Undokumentierte Typen von Schaltflächen
▼ Word-Interne Bezeichnung der Symbolleisten
▼ Symbolleisten schützen
▼ Links zu diesem Thema

Ausgangslage

▲ Zurück zum Anfang

> 1. Ich möchte ein von mir oft verwendetes Makro über eine Schaltfläche (einfacher Knopf) auf einer Symbolleiste starten können. Wie erreiche ich das?
>
> 2. Ich möchte ein benutzerdefiniertes *Listenkombinationsfeld* (Combo-Box) wie Word es z.B. für die Einstellung des Schriftgrades verwendet in einer Symbolleiste einbringen. Wie stelle ich das an?
>
> 3. Ich möchte ein benutzerdefiniertes (Unter-) Menü zusammenstellen. Wie packe ich das an?

Arten von Schaltflächen

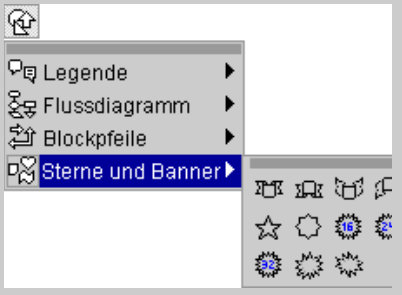
▲ Zurück zum Anfang

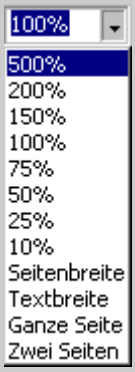
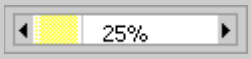
Word kennt eine ganze Reihe von verschiedensten Schaltflächenarten um dem Benutzer den Umgang mit der Anwendung zu vereinfachen. Allerdings können viele dieser Schaltflächen von Ihnen nicht erstellt oder verändert werden. Bei den wichtigsten und meistgebrauchten Schaltflächenarten ist dies aber durchaus möglich. Hier eine nicht ganz vollständige Zusammenstellung dieser Schaltflächenarten in Word.

Was beim Betrachten der Tabelle sicher auffallen wird, es gibt Schaltflächen, welche Sie sowohl über einen Word-Dialog, wie auch mittels VBA erstellen bzw. verändern können, andere wiederum sind nur mittels VBA manipulierbar und eine letzte Kategorie ist statisch. Wenn Sie wissen möchten, wie Sie die eine oder die andere Schaltfläche einrichten können, dann klicken Sie auf das entsprechende Bild in der Tabelle.

Bitte beachten Sie, dass zwischen der Menüleiste und den Symbolleisten eigentlich gar kein fundamentaler Unterschied besteht. Eigentlich ist die Menüleiste nichts anderes als eine Symbolleiste mit einer speziellen Erscheinungsform.

Schaltflächenvielfalt in MS-Office

Name	Interner Name	Erstellen über Word-Oberfläche	Erstellen mittels VBA	Beispiel
Einfacher Schaltknopf	Button	▼ Schaltknopf einrichten	▼ Mit VBA programmieren	
Nach unten aufklappbarer Schaltknopf	Button DropDown	nicht möglich	nicht möglich	
Aufklappbarer Schaltknopf	Button Popup	nicht möglich	nicht möglich	
Kombinationsfeld	ComboBox	nicht möglich	▼ Mit VBA Programmieren	

Name	Interner Name	Erstellen über Word-Oberfläche	Erstellen mittels VBA	Beispiel
DropDown Feld	DropDown	nicht möglich	▼ **Mit VBA möglich, allerdings nicht dokumentiert**	
Textfeld	Edit	nicht möglich	▼ **Mit VBA möglich, allerdings nicht dokumentiert**	
Fortschritts-leiste	Gauge	nicht möglich	nicht möglich	
Raster	Grid	nicht möglich	nicht möglich	
Graphisches Kombinations-feld	Graphic DropDown	nicht möglich	nicht möglich	

Name	Interner Name	Erstellen über Word-Oberfläche	Erstellen mittels VBA	Beispiel
Menü oder Untermenü	Popup	▼ **Menü oder Untermenü einrichten**	▼ **Mit VBA Programmieren**	

Einfacher Schaltknopf

Integrierter Word-Befehl mit einem einfachen Schaltknopf verknüpfen

▲ Zurück zum Anfang

Um einen integrierten Word-Befehl mit einem neuen Schaltknopf auf einer Ihren Symbolleisten zu verbinden, gehen Sie wie folgt vor:

1. Rufen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen...*** auf.
2. Wechseln Sie im nun erscheinenden Dialog in der Reiter ***Befehle***.
3. Markieren Sie in der linken Liste **Kategorien:** den Eintrag **Alle Befehle**.
4. Suchen Sie in der rechten Liste **Befehle:** Ihren Lieblingsbefehl aus.
5. Packen Sie diesen Befehl mit dem Mauszeiger und ziehen Sie ihn auf eine Symbolleiste Ihrer Wahl.
6. Immer noch im Anpassungsmodus, können Sie einen Rechtsklick auf den Knopf ausüben und im nun erschienen Kontextmenü, Aussehen und Symbol des Knopfes anpassen. Was für Möglichkeiten es gibt, das Aussehen eines Schaltknopfes zu verändern, erfahren Sie etwas weiter unten.
7. Abschliessend verlassen Sie diesen Dialog mit der Schaltfläche ***Schliessen.***.

Bestehendes Makro mit einem einfachen Schaltknopf verknüpfen

▲ Zurück zum Anfang

Um ein vorhandenes Makro mit einem neuen Schaltknopf auf einer Ihren Symbolleisten zu verbinden, gehen Sie wie folgt vor:

1. Rufen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen...*** auf.
2. Wechseln Sie im nun erscheinenden Dialog in der Reiter ***Befehle***.
3. Markieren Sie in der linken Liste **Kategorien:** den Eintrag **Makros**.
4. Suchen Sie in der rechten Liste **Befehle:** nach dem gewünschten Makro.
5. Packen Sie dieses Makro mit dem Mauszeiger und ziehen Sie ihn auf eine Symbolleiste Ihrer Wahl.
6. Immer noch im Anpassungsmodus, können Sie einen Rechtsklick auf den Knopf ausüben und im nun erschienen Kontextmenü das Aussehen des Knopfes beeinflussen. Insbesondere sollten Sie dem Makro einen einprägsameren Name und ein anders Symbol vergeben. Damit statt Text ein Symbol erscheint, wechseln Sie von *Nur Text* auf **Standard**.
7. Abschliessend verlassen Sie diesen Dialog mit der Schaltfläche ***Schliessen.***.

Erscheinungsbild eines Schaltknopfes beeinflussen

▲ Zurück zum Anfang

Im Anpassungsmodus (***Extras - Anpassen***) können Sie einen Rechtsklick auf eine Schaltfläche ausüben und dann das Aussehen des Schaltknopfes verändern. Folgende Optionen stehen Ihnen zur Verfügung:

- Im Feld **Name** können Sie der Schaltfläche einen aussagekräftigen Namen verpassen. Falls die Schaltfläche als Menüpunkt dargestellt wird. erscheint dieser Name 1:1 so wie sie ihn eingegeben haben. Wenn Sie diesen Menüpunkt nicht nur mit der Maus erreichen wollen sondern auch über eine Tastenkombination, dann verwenden Sie das **&**-Zeichen (kaufmännisches und) und zwar vor dem Zeichen, welches in der Tastenkombination enthalten sein soll. Dieses Zeichen wird dann unterstrichen dargestellt. Sie sollten darauf achten, dass pro Menü das selbe Zeichen nur einmal zur Anwendung kommt. Auf englisch nennt sich dieses Ding übrigens *Accelerator*. Hier ein Beispiel dazu: Sie können den Befehl mit dem Namen Suchen... im Menü Bearbeiten ohne die Maus zu benützen, also mit einer Tastenkombination wie folgt ansprechen: **[Alt + B + S]**.
  In VBA entspricht dies der Eigenschaft **Caption**.
- Mit den Befehlen **Schaltfläche kopieren** und **Schaltfläche einfügen** können Sie ein Sinnbild zwischen verschiedenen Schaltflächen austauschen. Sie können aber auch z.B. im Word-Dokument eine Grafik einfügen, diese dann in die Zwischenablage kopieren und anschliessend mit mit Befehl Schaltfläche einfügen diese Grafik auf eine Schaltfläche Ihrer Wahl bringen. Die Skalierung der Grafik wird Ihnen dabei abgenommen. Beachten Sie, dass diese Aktionen für eine Schaltfläche, welche auf *Nur Text* gesetzt ist, nicht zur Verfügung steht.
  In VBA entspricht dies der Methode **CopyFace** bzw. **PasteFace**.
- Mit dem Befehl **Schaltfläche zurücksetzen** können Sie das Sinnbild der Schaltfläche auf den Word-Standard zurücksetzen.
  In VBA entspricht dies der Methode **Reset**.
- Mit dem Befehl **Schaltflächebearbeiten...** tut sich ein kleiner Editor auf. Sie können in diesem Editor das Sinnbild der Schaltfläche verändern oder ein eigenes kreieren.
- Mit dem Befehl **Schaltflächensymbol ändern** können Sie ein Sinnbild für die Schaltfläche auswählen. Die Auswahl ist aber doch sehr beschränkt und Sie haben keine Möglichkeit diese Auswahl zu ergänzen. Der einzige Ort, an dem Sie ein eigens kreiertes Sinnbild aufbewahren können ist die Symbolleiste.
- Die nächsten vier Befehle in diesem Kontextmenü legen fest, ob die Schaltfläche ein Sinnbild oder Text enthalten soll. Dabei können Sie zwischen folgenden vier Optionen auswählen:
  - **Standard** Wenn sich der Befehl in einem Menü befindet, wird sowohl das Sinnbild, wie auch der Text angezeigt. Befindet sich der Befehl auf einer Symbolleiste wird nur das Sinnbild angezeigt.
  - **Nur Text (immer)** Gleichgültig, ob sich der Befehl in einem Menü oder in einer Symbolleiste befindet, es wird immer nur der Text angezeigt.
  - **Nur Text (in Menüs)** Befindet sich der Befehl in einem Menü dann wird nur der Text angezeigt, ist er dagegen in einer Symbolleiste wird nur das Sinnbild angezeigt.
  - **Schaltflächensymbol und Text** Gleichgültig, ob sich der Befehl in einem Menü oder in einer Symbolleiste befindet, es wird immer sowohl das Sinnbild wie auch der Text angezeigt. **Anmerkung:** Normalerweise sollten Sie mit der Einstellung **Standard** gut fahren:
    In VBA entspricht dies der Eigenschaft **Style**.
- Zum Schluss finden Sie noch die Option **Gruppierung beginnen**. Damit können Sie angeben, dass vor dem Befehl, welchen Sie gerade bearbeiten einen Trennstrich eingefügt werden soll.
  In VBA entspricht dies der Eigenschaft **BeginGroup**.

Kontext festlegen in dem die Symbolleiste verändert werden soll

Im Dialog ***Extras - Anpassen*** können sie im Kombinationsfeld mit dem Namen **Speichern in:** festlegen, ob die Veränderung der Symbolleiste ausschliesslich im aktiven Dokument oder in allen Dokumenten, welche auf Basis der aktiven Dokumentenvorlage erstellt werden wirksam werden soll. Als zusätzliche Option, können Sie dort auch die Einstellung **Normal** wählen. In diesem Falle geschieht die Veränderung in der globalen Vorlage mit dem Namen *Normal.dot* und wirkt sich dann generell aus. Sie sollten diese Einstellung beachten um wirklich zu Ihrem geplanten Ziel zu gelangen.

In VBA steuern Sie sie gleiche Angelegenheit übrigens mittels des sogenannten **CustomizationContext**.

Schaltknopf mittels VBA erstellen

▲ Zurück zum Anfang

Einfacher Schaltknopf Word global

Folgender Code erstellt auf der Symbolleiste **Standard** einen Schaltknopf, welcher den Windows-Rechner startet. Der Knopf wird in der Standardvorlage **Normal.dot** gespeichert und steht deshalb in der ganzen Word-Anwendung zur Verfügung. Die Routine **SchaltknopfEinrichten** muss grundsätzlich nur einmal ablaufen, wo gegen die Routine mit dem Namen **WindowsRechnerStarten** jedes mal aufgerufen wird, wenn der Schaltknopf betätigt wird.

```
Private Const Leiste = "Standard"

Sub SchaltknopfEinrichten()
  Application.CustomizationContext = NormalTemplate
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars(Leiste).Controls.Add(ID:=1)
  With Knopf
    .Caption = "Rechner"
    .TooltipText = "Windows-Rechner starten"
    .OnAction = "WindowsRechnerStarten"
    .FaceId = 283
  End With
  NormalTemplate.Save
End Sub

Sub WindowsRechnerStarten()
  Shell "calc.exe", vbMaximizedFocus
End Sub
```

(Erstellt und unterstützt einen einfachen Schaltknopf auf der Symbolleiste mit dem Namen "Standard".)

Einfacher Schaltknopf für eine bestimmte Vorlage

Folgender VBA-Code erstellt zuerst eine **neue Symbolleiste** mit dem Namen *Briefassistent*. Anschliessend wird auf dieser Symbolleiste ein Schaltknopf erstellt. Der Vorgang läuft immer ab, wenn basierend auf die Vorlage ein neues Dokument generiert wird. Symbolleiste und Schaltknopf stehen hier bewusst nur den Dokumenten zur Verfügung, welche auf Basis dieser Vorlage erstellt wurden.

```
Sub AutoNew()
  SchaltknopfEinrichten
End Sub

Private Sub SchaltknopfEinrichten()
  Application.CustomizationContext = Templates(1)
  Dim oLeiste As CommandBar
  Set oLeiste = CommandBars.Add(Position:=msoBarFloating, Temporary:=True)
  With oLeiste
    .Name = "Briefassistent"
    .Visible = True
  End With
  Dim Knopf As CommandBarButton
  Set Knopf = oLeiste.Controls.Add(ID:=1, Temporary:=True)
  With Knopf
    .Caption = "Dein Briefassistent"
    .TooltipText = "Stertet Deinen Briefassistenten"
    .OnAction = "BriefassistentStarten"
    .FaceId = 59
    .Style = msoButtonIconAndCaption
  End With
  Templates(1).Saved = True
End Sub

Sub BriefAssistentStarten()
  MsgBox "Hier kommt Dein Briefassistent."
End Sub
```

(Erstellt eine neue Symbolleiste mit einem einfachen Schaltknopf. Die Symbolleiste mit dem Schaltknopf steht bewusst nur jenen Dokumenten zur Verfügung, welche auf Basis der Vorlage, welche Trägerin des Codes ist, erstellt werden.)

Ein-/Ausschalter (Toggle-Button)

▲ Zurück zum Anfang

Im folgenden wird auf der Symbolleiste **Standard** ein Schaltknopf erstellt, welcher die Funktion eines **Ein-/Ausschalter** übernehmen kann. Ist die **Formatvorlagenanzeige** ausgeschaltet, wird diese beim Klick auf den Knopf eingeschaltet. Der Knopf ändert dabei sein Aussehen, er ist eingerastet. Beim nochmaligen Betätigen des Knopfes, wird die Formatvorlagenanzeige wieder ausgeblendet und der Knopf erlangt sein ursprüngliches Aussehen zurück. Die Routine **ToggleKnopfEinrichten** muss grundsätzlich nur einmal ablaufen, wogegen der Prozess mit dem Namen **ToggleKnopfProzess** jedes mal aufgerufen wird, wenn der Schaltknopf vom Benutzer betätigt wirf.

```
Sub ToggleKnopfEinrichten()
  Application.CustomizationContext = NormalTemplate
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars("Standard").Controls.Add(ID:=1)
  With Knopf
    .Caption = "Formatvorlagen einblenden"
    .TooltipText = "Blendet im linken Bereich die Namen der Formatvorlagen ein."
    .OnAction = "ToggleKnopfProzess"
    .FaceId = 2174
  End With
  NormalTemplate.Save
End Sub

Sub ToggleKnopfProzess()
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars.ActionControl
  With Knopf
    If ActiveWindow.StyleAreaWidth = 0 Then
      ActiveWindow.StyleAreaWidth = CentimetersToPoints(1.8)
      .State = msoButtonDown
      .Caption = "Formatvorlagen ausblenden"
      .TooltipText = "Blendet die Namen der Formatvorlagen aus."
    Else
    ActiveWindow.StyleAreaWidth = 0
      .State = msoButtonUp
      .Caption = "Formatvorlagen einblenden"
      .TooltipText = "Blendet die Namen der Formatvorlagen ein."
    End If
  End With
End Sub
```

(Erstellt und unterstützt einen Ein/-Ausschaltknopf auf der Symbolleiste mit dem Namen "Standard".)

Das Ganze hat übrigens einen Schönheitsfehler. Wird die Formatvorlagenanzeige statt über diese Schaltfläche über den ordentlichen Word-Dialog ein- oder ausgeschaltet, dann verändert sich das Aussehen des Knopfes leider nicht dynamisch. Der Knopf synchronisiert sich diesbezüglich erst wieder, wenn er das nächste mal betätigt wird. Die word-integrierten Ein-/Ausschalter haben diesen Makel nicht. Mit Word2000 ist es möglich, Ereignisse besser zu handhaben als dies für die Vorgängerversionen von Word zutrifft. Mit zusätzlichem Code ist dort dieses mit viel Aufwand Manko auszugleichen.

Nachfolgend eine Tabelle, welche aufzeigt, welche Eigenschaften und Methoden Sie verwenden können um Aussehen und Verhalten von Schaltknöpfen zu beeinflussen:

Wichtige Eigenschaften und Methoden für Schaltknöpfe

▲ Zurück zum Anfang

Eigenschaft oder Methode	Erklärung
Caption	Legt die **Bezeichnung** des Schaltknopfes fest
Style	Legt fest, ob der Schaltknopf als Text, als Sinnbild oder als Kombination von beidem erscheinen soll
BeginGroup	Legt fest, ob vor dem Schaltknopf eine **Trennlinie** eingefügt werden soll
Visible	Legt fest, ob der Schaltknopf angezeigt wird
Enabled	Legt fest, ob der Schaltknopf angesprochen werden kann oder grau erscheint
State	Legt fest, ob der Schaltknopf ein- oder ausgerastet angezeigt wird.
FaceID	Legt die Nummer des **Sinnbildes** fest, welches den Schaltknopf zieren soll. Wie Sie eine Liste aller verfügbaren Sinnbilder mit ihrer ID enthalten finden Sie im folgenden Skript: ► **Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen?**
TooltipText	Legt den Text fest, welcher erscheint, wenn der Benutzer einige Zeit mit dem Zeiger über dem Schaltknopf verharrt. Diese Information wird auch Quick-Info genannt und ist in der Regel zitronengelb hinterlegt.
OnAction	Hier geben Sie an, welcher Prozess (**Makro**) ausgeführt werden soll, wenn der Benutzer den Schaltknopf betätigt.
Tag	Hier können Sie der Schaltfläche eine eindeutige Kennung verpassen, welche Sie z.B. zum Zeitpunkt auslesen können, wenn das Makro aufgerufen wird, welches Sie mit der Eigenschaft OnAction gesetzt haben. Die VBA-Methode FindControl erlaubt es auch, mittels dieser Kennung einen Schaltknopf über alle Symbolleisten zu finden. Für die Eindeutigkeit der Kennung (also des Tags) müssen Sie selbst besorgt sein. In der Praxis ist es besser, einen Schaltknopf über diese Eigenschaft zu suchen als über die Eigenschaft Caption, nicht zuletzt deshalb, weil die Tag-Eigenschaft nicht sprachspezifisch ist. Der Einsatz der FindControl-Methode wird später noch ausführlich behandelt.
CopyFace	Mit dieser Methode können Sie das Sinnbild des Schaltknopfes in die Zwischenablage kopieren.
PasteFace	Mit dieser Methode können Sie das Sinnbild, welches sich gerade in der Zwischenablage befindet, einem Schaltknopf zuweisen.

Kombinationsfelder (ComboBoxes)

▲ Zurück zum Anfang

Kombinationsfelder lassen sich ausschliesslich mittels VBA erstellen. Folgende beiden Beispiele generieren ein Kombinationsfeld auf der Symbolleiste **Standard** in der Standardvorlage **Normal.dot**. Das Steuerelement soll eine Liste von Dokumentenvorlagen enthalten und es dem Benutzer ermöglichen eine Auswahl zu treffen. Wählt der Benutzer in diesem Kombinationsfeld eine Vorlage aus, wird auf Basis dieser Vorlage ein neues Dokument generiert..

Das erste Beispiel sieht eine statische Lösung vor. Die Dokumentenvorlagen, welche Sie zur Auswahl stellen möchten, werden im Kopf des VBA-Codes explizit von Ihnen eingetragen.

Die zweite Lösung ist dynamisch. Der Prozess, welche das Kombinationsfeld aufbaut, startet jedes Mal, wenn Word gestartet wird. Damit ist die Liste der Vorlagen auch ohne Ihr Zutun immer auf dem aktuellen Stand. Die Routine ermittelt zuerst das Vorlagenverzeichnis, liest anschliessend die Namen aller Word-Dokumentenvorlage in diesem Verzeichnis ein und füllt anschliessend diese Namen in das Steuerelement ein.

Statische Lösung

Um das folgende Code-Beispiel zu testen, gehen Sie wie folgt vor:

1. Um das Kombinationsfeld einzurichten kopieren Sie den Prozess mit dem Namen **ComboBoxVorlageEinrichtenStatisch** im VBA-Editor in ein neues Modul.
2. Passen Sie die Namen der Dokumentenvorlagen in den Variablen **V(1)** bis **V(n)** Ihren Bedürfnissen an.
3. Passen Sie den Wert in der Variablen **AnzVorlagen** an. Sie müssen dort angeben, wie viele Vorlagen es im Gesamten sind.
4. Führen Sie nun das Makro aus. Grundsätzlich muss dieses Makro genau einmal ablaufen, es sei denn, Sie fügen in der Liste neue Vorlagen hinzu oder wollen das Erscheinungsbild des Kombinationsfeldes ändern. Das Makro stellt auch sicher, dass das Kombinationsfeld von einem früheren Versuch gelöscht wird.
5. Lassen Sie sich jetzt im VBA-Editor mit der Tastenkombination **[Strg + R]** den Projekt-Explorer anzeigen.
6. Markieren Sie im Projekt-Explorer den Eintrag **Normal** und wählen Sie im Menü den Dialog ***Einfügen - Modul***.
7. Kopieren Sie die Ereignisprozedur mit dem Namen **ComboBoxVorlageChange** in das leere Fenster, welches sich geöffnet hat. Dieser Prozess wird jedes Mal ausgeführt, wenn der Benutzer im Kombinationsfeld eine Einstellung vornimmt. Der Code der Ereignisprozedur muss **nicht** geändert werden, falls Sie dem Kombinationsfeld neue Vorlagen hinzufügen.

```
Private Const Leiste = "Standard"
Private Const Breite = 4
Private Const Tag = "SW19990927"
Private Const EreignisProzedur = "ComboBoxVorlageChange"

Sub ComboBoxVorlageStatischEinrichten()
  Pfad = Options.DefaultFilePath(wdUserTemplatesPath)
  AnzVorlagen = 5
  ReDim V(AnzVorlagen)
  V(1) = "Normal"
  V(2) = "Geschäftsbrief"
  V(3) = "Einladung"
  V(4) = "Party"
  V(5) = "Rechnung"
  CustomizationContext = NormalTemplate
  Dim SL As CommandBar
  Dim CB As CommandBarComboBox
  Set CB = CommandBars.FindControl(Type:=msoControlComboBox, Tag:=Tag)
  If Not CB Is Nothing Then CB.Delete
  Set CB = CommandBars(Leiste).Controls.Add(Type:=msoControlComboBox)
  With CB
    .Tag = "SW19990927"
    .BeginGroup = True
    .Caption = "Vorlage auswählen"
    .TooltipText = "Erstellt ein neues Dokument auf Basis der gewählten Vorlage."
    .OnAction = EreignisProzedur
    .Width = CentimetersToPoints(Breite)
    .DropDownWidth = CentimetersToPoints(Breite)
    .DropDownLines = AnzVorlagen
    For i = 1 To AnzVorlagen
      .AddItem V(i)
    Next i
    .ListHeaderCount = 3
    .ListIndex = 0
  End With
  NormalTemplate.Save
End Sub
```

(Richtet ein Kombinationstextfeld [ComboBox] auf der Symbolleiste *Standard* ein. Die Ereignisprozedur dazu, soll den Namen "ComboBoxVorlageChange" tragen.)

```
Private Sub ComboBoxVorlageChange()
  Titel = "Dokument erstellen"
  Pfad = Options.DefaultFilePath(wdUserTemplatesPath)
  Vorlage = CommandBars.ActionControl.Text
  Q = Chr(34)
  On Error Resume Next
  Documents.Add Template:=Pfad & "\" & Vorlage
  Select Case Err.Number
    Case 0
    Case 5151
      MsgBox "Die Vorlage " & Q & Pfad & "\" & Vorlage & ".dot" & Q & _
         "existiert nicht.", vbCritical, Titel
    Case Else
      MsgBox "Unerwarteter Fehler: " & Err.Number & vbCrLf & _
         Err.Description, vbCritical, Titel
  End Select
  On Error GoTo 0
End Sub
```

(Stellt die Ereignisprozedur zum beschriebenen Kombinationsfeld dar.)

Dynamische Lösung

Hier nun also die dynamische Lösung. Alle Word-Vorlagen im Standardverzeichnis für Vorlagen werden eingelesen, um diese in einem Kombinationsfeld dem Benutzer zur Wahl zu stellen. Werden im Laufe der Zeit neue Vorlagen erstellt, synchronisiert sich diese Routine selbständig. Die Ereignis-Routine, welche auf die Wahl reagiert, bleibt sich gleich wie im obigen Beispiel. Deshalb wird diese hier nicht noch einmal abgedruckt.

Um das folgende Code-Beispiel zu testen, gehen Sie wie folgt vor:

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot**.
2. Wechseln Sie in die VBA-Umgebung und kopieren Sie die beiden Prozesse mit den Namen **ComboBoxVorlageEinrichtenDynamisch** und **ComboBoxVorlageChange** in ein neues Modul.
3. Fügen Sie in die **Normal.dot** über ***Einfügen - Modul*** ein neues Modul ein.
4. Kopieren Sie den Prozess mit dem Namen **AutoExec** in dieses Modul. Der Name des Prozesses (AutoExec) ist dabei zwingend, wogegen der Name des Moduls irrelevant ist.
5. Speichern Sie die "Normal.dot" und testen Sie die Sache.

```
Sub AutoExec
  ComboBoxVorlageEinrichtenDynamisch
End Sub
```

(Der Aufruf der Routine, welche das Kombinationsfeld einrichtet erfolgt aus einem Prozess mit dem Namen *AutoExec*. Damit ist sichergestellt, dass der Prozess jedes Mal wenn Word startet, aufgerufen wird.)

```
Private Const Leiste = "Standard"
Private Const Breite = 4
Private Const Tag = "SW19990927"
Private Const EreignisProzedur = "ComboBoxVorlageStatischChange"

Sub ComboBoxVorlageEinrichtenDynamisch()
  Pfad = Options.DefaultFilePath(wdUserTemplatesPath)
  Dim V() As String
  chk = Dir(Pfad & "\*.dot")
  i = -1
  While chk <> ""
    If Len(chk) > 4 And Left(chk, 1) <> "~" Then
      i = i + 1
      ReDim Preserve V(i)
      V(i) = Left(chk, Len(chk) - 4)
    End If
    chk = Dir
  Wend
  If i > 1 Then WordBasic.SortArray V()
  AnzVorlagen = i + 1
  CustomizationContext = NormalTemplate
  Dim SL As CommandBar
  Dim CB As CommandBarComboBox
  Set CB = CommandBars.FindControl(Type:=msoControlComboBox, Tag:=Tag)
  If Not CB Is Nothing Then CB.Delete
  Set CB = CommandBars(Leiste).Controls.Add(Type:=msoControlComboBox)
  With CB
    .Tag = Tag
    .BeginGroup = True
    .Caption = "Vorlage auswählen"
    .TooltipText = "Erstellt ein neues Dokument auf Basis der gewählten Vorlage."
    .OnAction = "ComboBoxVorlageChange"
    .Width = CentimetersToPoints(Breite)
    .DropDownWidth = CentimetersToPoints(Breite)
    .DropDownLines = AnzVorlagen
    For i = 0 To AnzVorlagen - 1
      .AddItem V(i)
    Next i
    .ListHeaderCount = 3
    .ListIndex = 0
  End With
  NormalTemplate.Save
End Sub
```

(Richtet ein Kombinationsfeld [ComboBox] auf der Symbolleiste *Standard* ein. Die Liste wird dynamisch aufgebaut. Die Ereignisprozedur dazu, soll den Namen "ComboBoxVorlageChange" tragen. Diese Ereignisprozedur finden Sie gleich anschliessend.)

```
Private Sub ComboBoxVorlageChange()
  Titel = "Dokument erstellen"
  Pfad = Options.DefaultFilePath(wdUserTemplatesPath)
  Vorlage = CommandBars.ActionControl.Text
  Q = Chr(34)
  On Error Resume Next
  Documents.Add Template:=Pfad & "\" & Vorlage
  Select Case Err.Number
    Case 0
    Case 5151
      MsgBox "Die Vorlage " & Q & Pfad & "\" & Vorlage & ".dot" & Q & _
         "existiert nicht.", vbCritical, Titel
    Case Else
      MsgBox "Unerwarteter Fehler: " & Err.Number & vbCrLf & _
         Err.Description, vbCritical, Titel
  End Select
  On Error GoTo 0
End Sub
```

(Stellt die Ereignisprozedur zum beschriebenen Kombinationsfeld dar.)

Angenommen, Sie möchten dieses Kombinationsfeld nur für Dokumente einrichten, welche auf einer ganz bestimmten Vorlage beruhen, müssen Sie die Sache etwas anders einrichten. In diesem Falle gehen Sie wie folgt vor:

1. **Öffnen** Sie die Dokumentenvorlage Ihrer Wahl.
2. Wechseln Sie in die VBA-Umgebung und kopieren Sie die beiden Prozesse mit den Namen **ComboBoxVorlageEinrichtenDynamisch** und **ComboBoxVorlageChange** in ein neues Modul. Sie finden die beiden Prozesse etwas weiter oben. (Sie bleiben sich auch in dieser Implementierungsvariante gleich).
3. Fügen Sie in diese Vorlage über ***Einfügen - Modul*** ein neues Modul ein.
4. Kopieren Sie den Prozess mit dem Namen **AutoNew** in dieses Modul. Der Name des Prozesses (AutoNew) ist dabei zwingend, wogegen der Name des Moduls irrelevant ist.
5. Speichern Sie die Vorlage und testen Sie die Sache.

```
Sub AutoNew
  ComboBoxVorlageEinrichtenDynamisch
End Sub
```

(Der Aufruf der Routine, welche das Kombinationsfeld einrichtet erfolgt aus einem Prozess mit dem Namen *AutoNew* Damit ist sichergestellt, dass jedes Mal, wenn ein neues Dokument auf Basis der Dokumentenvorlage erstellt wird, der Prozess gestartet wird.)

Nachfolgend eine Tabelle, welche aufzeigt, welche Eigenschaften und Methoden Sie verwenden können um Aussehen und Verhalten von mit VBA erstellen Kombinationsfeldern zu beeinflussen:

Wichtige Eigenschaften und Methoden für Kombinationsfelder

▲ Zurück zum Anfang

Eigenschaft oder Methode	Erklärung
Caption	Legt die **Bezeichnung** des Kombinationsfeldes fest
Style	Legt fest, ob das Kombinationsfeld mit oder ohne Bezeichnung auf der linken Seite erscheint. Gültige Werte sind: msoComboLabel oder msoComboNormal. Wenn Sie hier msoComboLabel wählen, wird die Caption-Eigenschaft vorne angestellt.
BeginGroup	Legt fest, ob vor dem Kombinationsfeld eine **Trennlinie** eingefügt werden soll
Visible	Legt fest, ob das Kombinationsfeld angezeigt wird
Enabled	Legt fest, ob das Kombinationsfeld angesprochen werden kann oder grau erscheint
Width	Legt die Breite des Kombinationsfeldes fest und zwar in Pixel. Möchten Sie ein Kombinationsfeld von einer Breite von 2 cm, codieren Sie folgendes: .Width = CentimetersToPoints(2)
DropDownWidth	Legt die Breite des erscheinenden Liste fest. Diese Breite kann also anders gewählt werden, als die Breite der Schaltfläche selbst.
DropDownLines	Legt fest, wie viele Einträge angezeigt werden sollen, wenn das Kombinationsfeld aufklappt. Wird hier der Wert 0 gesetzt, entspricht dies der Zahl der Einträge in der Liste. Das scrollen für den Benutzer erübrigt sich dann in diesem Falle.
AddItem	Verwenden Sie diese Methode, um dem Kombinationsfeld einen Eintag hinzuzufügen. Verwenden Sie pro Eintrag ein **AddItem**-Satement.
ListCount	Gibt die **Anzahl der Einträge** zurück, welche das Kombinationsfeld enthält. Sie brauchen und können diesen Wert nicht explizit zu setzen. Er ergibt sich aus der Anzahl der Einträge, welche Sie dem Listenfeld hinzufügen.
List(index)	Gibt den Wert eines bestimmten Listeneintrages des Kombinationsfeldes zurück. Die Zählung beginnt mit 1.
ListIndex	Gibt die Nummer des markierten Eintrages im Kombinationsfeld zurück oder legt diesen fest. Die Zählung der Listeneinträge beginnt mit 1. Wenn der Benutzer im Kombinationsfeld eine neue Auswahl trifft, ändert sich der ListIndex und das Makro, welches Sie in der Eigenschaft OnAcion gewählt haben, wird ausgeführt. Das selbe tritt ein, wenn Sie im Code den ListIndex verändern, also neu setzen. Schauen Sie sich dazu auch das Kapitel, welche die ActionControl-Eigenschaft bespricht in diesem Skript an.
ListHeaderCount	Legt die Anzahl der Listeneinträge, welche über der Trennlinie angezeigt werden sollen fest. Dabei handelt es sich um die zuletzt gewählten Einträge. Wenn Sie keine Trennlinie möchten, setzen Sie diesen Wert auf **-1** (also minus 1). Sie können diese Eigenschaft erst setzen, nachdem Sie dem Kombinationsfeld Einträge zugefügt haben.
TooltipText	Legt den Text fest, welcher erscheint, wenn der Benutzer einige Zeit mit dem Zeiger über dem Kombinationsfeld verharrt. Diese Information wird auch **Quick-Info** genannt und ist in der Regel zitronengelb hinterlegt.
OnAction	Hier geben Sie an, welcher Prozess (**Makro**) ausgeführt werden soll, wenn der Benutzer eine Auswahl im Kombinationsfeld vornimmt.
Tag	Hier können Sie dem Kombinationsfeld eine eindeutige Kennung verpassen, welche Sie z.B. zum Zeitpunkt auslesen können, wenn das Makro aufgerufen wird, welches Sie mit der Eigenschaft OnAction gesetzt haben. Die VBA-Methode FindControl erlaubt es auch, mittels dieser Kennung ein Kombinationsfeld über alle Symbolleisten zu finden. Für die Eindeutigkeit der Kennung (also des Tags) müssen Sie selbst besorgt sein. In der Praxis ist es besser, ein Kombinationsfeld über diese Eigenschaft zu suchen als über die Eigenschaft Caption, nicht zuletzt deshalb, weil die Tag-Eigenschaft nicht sprachspezifisch ist. Der Einsatz der FindControl-Methode wird später noch ausführlich behandelt.
Text	Den Eintrag (Text), welche der Benutzer im Kombinationsfeld gewählt hat.

Menü-Leiste

Menüs können verschachtelt werden, das heisst ein Menü hat Untermenüs, welches seinerseits wieder Untermenü aufweisen kann, bis ein ganzer verästelter Baum entsteht. Sie können dem Hauptmenü, einem Untermenü oder einer Symbolleiste ein Menü zufügen etc.

Neues Menü über Word-Dialog erstellen

▲ Zurück zum Anfang

Um ein neues (Unter-)Menü im Hauptmenü oder auf einer Symbolleiste Ihrer Wahl zu erstellen, gehen Sie wie folgt vor:

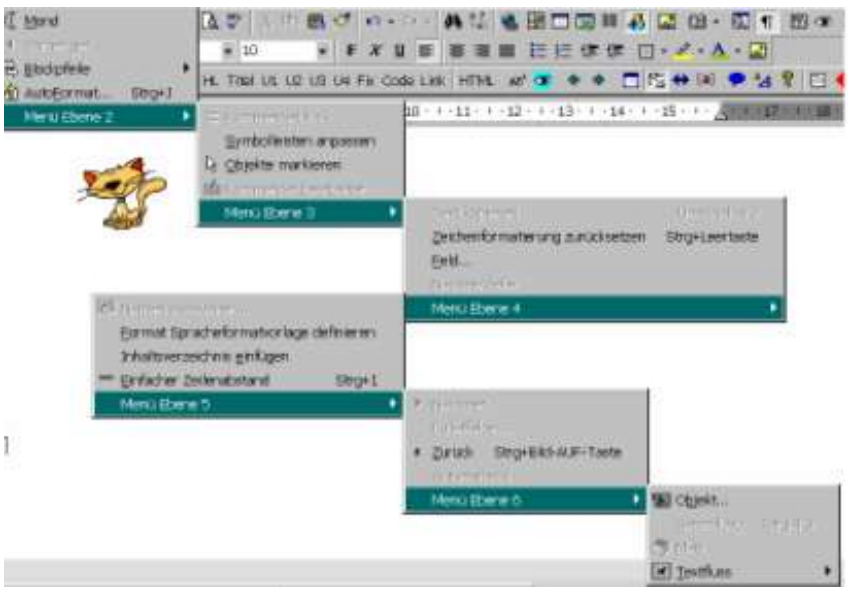
1. Rufen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen...*** auf.
2. Wechseln Sie im nun erscheinenden Dialog in den Reiter ***Befehle***.
3. Markieren Sie in der linken Liste **Kategorien:** den Eintrag **Neues Menü**.
4. Packen Sie auf der rechten Liste **Befehle:** den Eintrag **Neues Menü** mit dem Mauszeiger und ziehen Sie ihn auf die Menüleiste oder auf eine Symbolleiste Ihrer Wahl. Sie können auch ein neues Untermenü in einem Untermenü erstellen. Sobald Sie mit dem Zeiger auf einem Menü sind, klappt dieses auf.
5. Immer noch im Anpassungsmodus, führen Sie einen Rechtsklick auf das Menü aus und passen Name und Aussehen des neuen Menüs Ihren Bedürfnissen an.
6. Nun können Sie das erstelle Menü mit Befehlen oder Makros Ihrer Wahl füllen. Verwenden Sie dabei das selbe Vorgehen, wie es schon im Kapitel **Bestehender Word-Befehl mit einem einfachen Schaltknopf verknüpfen** beschrieben wurde.
7. Abschliessend verlassen Sie diesen Dialog mit der Schaltfläche ***Schliessen.***.

Sie können nun diesem Menü entweder eine beliebige Zahl von Befehlen zufügen, oder aber auch ein weiteres (Unter-)menü. Ziehen Sie die entsprechenden Schaltflächen im Anpassungsmodus einfach auf die neu erstellte Menüleise.

Neues Menü mit VBA erstellen

▲ Zurück zum Anfang

Das Kaskadieren von Menüelementen



Wie schon erwähnt können Sie Menüs kaskadieren. Diese Tatsache muss sich auch im Code wiederspiegeln. Das neu erstellte Menü ist immer Element des ihm übergeordnete Objektes. Das oberste Objekt in der Hierarchie ist eine Symbolleiste bzw. die Symbolleiste mit dem Namen *Menüleiste*, welche ein spezielles Aussehen aufweist und meist Hauptmenü genannt wird. Ein weiteres Untermenü ist dann Element eines bereits erstellten Menüs. Diese Kaskade können Sie dann weiter treiben bis zum geht nicht mehr.

Das nachfolgende Beispiel mag in der Praxis nicht viel Sinn machen, aber es zeigt sehr schön auf, wie Menüs verschachtelt werden können. Daran anschliessend folgen dann noch praxisnahere und einfachere Beispiele.

```
Private Const Leiste = "Menu Bar"
Private Const Tag = "SW20000317"

Sub MenüKaskade()
  AnzKaskade = 6
  AnzBefehle = 4
  ReDim Menü(AnzKaskade) As CommandBarPopup
  Dim SF As CommandBarControl
  Rnd 'Zufallsgenerator initalisieren
  CustomizationContext = ActiveDocument
  Set Menü(1) = CommandBars.FindControl(Type:=msoControlPopup, Tag:=Tag)
  If Not Menü(1) Is Nothing Then Menü(1).Delete 'Aufräumen vom letzten Versuch
  For i = 1 To AnzKaskade
    If i = 1 Then 'Erste Ebene soll das Hauptmenü sein
      Set Menü(i) = CommandBars(Leiste).Controls.Add(Type:=msoControlPopup)
    Else
      Set Menü(i) = Menü(i - 1).Controls.Add(Type:=msoControlPopup)
    End If
    With Menü(i)
      If i = 1 Then .Tag = Tag
      .Caption = "Menü Ebene " & i
      .TooltipText = "Das ist das Menü Ebene " & i
      While Menü(i).Controls.Count < AnzBefehle
        CmdId = Int((3400 * Rnd) + 1) 'Zufallsbefehl ermitteln
        On Error Resume Next 'nicht alle Befehls-Id's existiern
        .Controls.Add ID:=CmdId
        On Error GoTo 0
      Wend
    End With
  Next i
End Sub
```

(Nicht *menu à la carte*, sondern Menü Zufall. Spielerischer Umgang mit Menüs und Untermenüs. Ein 6-fach verschachteltes Menü wird generiert und mit Befehlen gefüllt, welche nach dem Zufallsprinzip ermittelt werden.)

Der Menüleiste ein neues Untermenü zufügen

Im folgenden Beispiel wird an dritter Position der Menüleiste ein neues (Unter-)Menü hinzugefügt. Dieses Menü wird im selben Code mit drei einfachen Schaltknöpfen bestückt, welche je ein Aktion auslösen.

Sollte dieses Menü bereits von einem vorhergehenden Versuch existieren, wird zuerst aufgeräumt. Dazu macht sich der Code die **Tag-Eigenschaft** und die **FindControl-Methode** zu Nutze. Diese Mimik ist gegen Ende des Skriptes noch genauer erklärt.

Das Menü selbst ist Element der Symbolleiste **Menüleiste**, wogegen die Schaltknöpfe dann bereits Elemente des erstellten Menüs sein werden.

Die Änderungen passieren in der Normal.dot `CustomizationContext = NormalTemplate` und wirken dadurch global. Alternativen dazu wären: `CustomizationContext = ActiveDocument` oder `CustomizationContext = ActiveDocument.AttachedTemplate`.

```
Private Const SL = "Menu Bar"
Private Const Tag = "SW20000515"

Sub NeuesMenüErstellen()
  Dim Menü As CommandBarPopup
  Dim Knopf As CommandBarButton
  Set Menü = CommandBars.FindControl(Tag:=Tag)
  If Not Menü Is Nothing Then Menü.Delete
  CustomizationContext = NormalTemplate
  Set Menü = CommandBars(SL).Controls.Add(Type:=msoControlPopup, Before:=3)
  With Menü
    .Tag = Tag
    .Caption = "Allerlei"
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .Caption = "Macht Lärm"
      .OnAction = "Lärm"
      .TooltipText = "Macht nur ein wenig Lärm"
      .FaceId = 273
    End With
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .Caption = "Rechner"
      .OnAction = "Calculator"
      .TooltipText = "Ich kann's nicht im Kopf"
      .FaceId = 283
    End With
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .Caption = "Solitär"
      .OnAction = "Spielen"
      .TooltipText = "Mach mal Pause"
      .FaceId = 286
    End With
  End With
  NormalTemplate.Save
End Sub

Sub Lärm()
  Beep
End Sub

Sub Calculator()
  Shell "calc.exe"
End Sub

Sub Spielen()
  Shell "sol.exe", vbNormalFocus
End Sub
```

(Diese Routine erstellt ein neues Untermenü in der Symbolleiste *Menüleiste* (also im Hauptmenü). Das neue Menü wird dann mit drei funktionellen Schaltknöpfen bestückt.)

Einem Untermenü eine zusätzlich Schaltfläche zufügen

Die folgende Routine ist zur Abwechslung mal für **Excel** gedacht, denn beim *CommandBar-Objekt* handelt es sich um eine Office-weite Einrichtung. Als erstes wird auf der Menüleiste nach dem Untermenü (*PopUp-Menü*) **Datei** gesucht. Diese Control besitzt Die Befehls-ID **30002**. Innerhalb dieses Untermenüs wird dann nach dem Befehl **Drucken...** gesucht. Die Befehls-ID lautet hier **4** Anschliessend an diesen Befehle soll nun ein neuer Befehl mit dem Namen **Export nach GIF** erstellte werden. Diesem Befehl wird der Prozess mit dem Namen **ExportNachGIF** zugewiesen, welcher bewirkt, dass ein markiertes Diagramm im Excel-Tabellenblatt, als GIF-Grafik exportiert wird. Sollte der Befehl mit dem Namen *Drucken...* sich dummerweise nicht im Menü mit dem Namen Datei befinden, wird der neue Befehl am Ende eingefügt.

Erstellen Sie also eine neue Excel-Arbeitsmappe mit einem Tabellenblatt, welches mindestens ein Diagramm enthält. Kopieren Sie den Code, welcher unten abgedruckt ist im VBA-Editor in die Arbeitsmappen und lassen Sie die Routine mit dem Namen **ExportNachGIFEinrichten** genau einmal ablaufen. Damit ist der neue Schaltknopf eingerichtet.

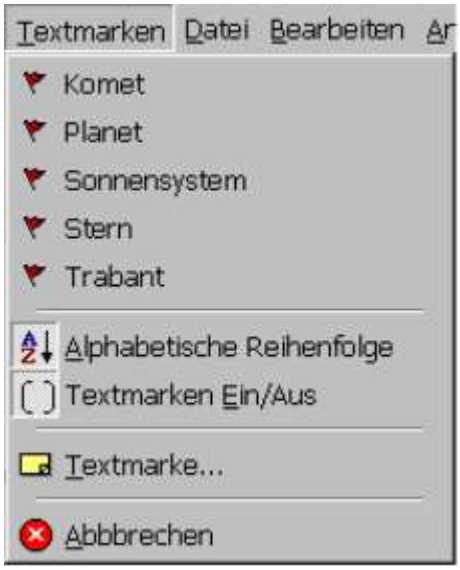
Der Prozess mit dem Namen **ExportNachGIF** wird jedes mal aufgerufen, wenn der Benutzer den neuen Befehl über das nun erweiterte Datei-Menü aufruft.

```
Sub ExportNachGIFEinrichten()
  Dim MenüLeiste As CommandBar
  Dim Menü As CommandBarPopup
  Dim Knopf As CommandBarButton
  Set MenüLeiste = CommandBars("Worksheet Menu Bar")
  Set Menü = MenüLeiste.FindControl(Type:=msoControlPopup, ID:=30002)
  Position = Menü.Controls.Count
  Dim x As CommandBarControl
  For Each x In Menü.Controls
    If x.ID = 4 Then
      Position = x.Index + 1
      Exit For
    End If
  Next
  Set Knopf = Menü.Controls.Add(Type:=msoControlButton, Before:=Position)
  With Knopf
    .Caption = "Export nach GIF"
    .TooltipText = "Exportiert ein Diagramm als GIF-Datei"
    .FaceId = 479
    .OnAction = "ExportNachGIF"
  End With
End Sub

Sub ExportNachGIF()
  Pfad = "c:\Eigene Dateien"
  If TypeName(Selection) <> "ChartArea" Then
    MsgBox "Es ist kein Diagramm markiert."
    End
  End If
  Dim Diag As Chart
  Set Diag = ActiveChart
  Datei = Pfad & "\Diag" & Diag.Name & ".GIF"
  Diag.Export Filename:=Datei, FilterName:="GIF"
  MsgBox "Das Diagramm wurde unter dem Namen " & Datei & " exportiert."
End Sub
```

(Richtet eine neue Schaltfläche im Menü *Datei* ein, welche es erlaubt, ein markiertes Diagramm in eine GIF-Datei zu exportieren.)

Zugriff auf Textmarken in Word übersichtlicher organisieren

▲ Zurück zum Anfang



Haben Sie sich auch schon geärgert, dass das Anspringen von Textmarken in einem Word-Dokument über den offiziellen Dialog umständlicher gelöst ist, also es wirklich sein müsste?

Wenn ja, dann dürfte es Sie interessieren, dass es ein sehr schönes Tool von der Firma ***PRIME*** gibt, welches Sie gratis aus dem Internet herunterladen können und welches dieses Manko ausgleicht. Der Link um dieses Utility, welches übrigens auch von Microsoft empfohlen wird und welches sich für Word 97 und Word2000 eignet, ist unten eingeblendet.

Der VBA-Code zu diesem Projekt ist übrigens Passwort-geschützt. Da es uns aber hier primär darum geht, etwas zu lernen, wollen wir versuchen, dieses Tool nachzubauen. Also *cracking by re-engineering* was zwar nicht besonders originell, dafür aber legal und legitim und oben drauf lehrreich ist.

Zudem kann die hier abgedruckte Lösung die Textmarken wahlweise sortiert nach dem Textmarkennamen oder sortiert nach der Reihenfolge, wie diese im Text erscheinen, anzeigen. Das Makro, welches die Firma *PRIME* zur Verfügung stellt, kann letzteres nicht.

Unser Tool unterstützt somit folgende Funktionen:

- Beim Öffnen des *PopUp*-Menüs wird das Dokument nach Textmarken durchsucht. Jede der gefundenen Testmarken ergibt einen Eintrag in unserem *PopUp*-Menü.
- Wird ein solcher Eintrag später vom Benutzer geklickt, wird die entsprechende Textmarke im Dokument angesprungen und das Menü schliesst sich wieder.
- Eine zusätzlich und separate Schaltfläche in diesem Menü, ermöglicht es uns, die Anzeige der Textmarken im Dokument entweder anzuzeigen oder. auszublenden.
- Dank einer weiteren Schaltfläche, können Sie die Sortierreihenfolge wie die Textmarken in diesem PopUp-Menü angezeigt werden, beeinflussen.
- Zusätzlich kann auch noch der Word-Integrierte Dialog **Textmarken...** aus diesem Pop-Up Menü aufgerufen werden.

Unser Pop-Up Menü wird auf der Hauptmenüleiste von Word erstellt und wird in der **Normal.dot** gespeichert. Beides können Sie aber im Kopf des Codes anpassen. Zudem steht es Ihnen frei anschliessend dieses Menü über ***Extras - Anpassen*** zu verschieben und weiter Ihren Bedürfnissen anzupassen.

Kopieren Sie den unten stehenden Code in ein neues Modul der Standardvorlage mit dem Namen **Normal.dot**. Der Prozess mit dem Namen **TextmarkenMenüEinrichten** muss grundsätzliche nur einmal ablaufen, wogegen die Prozesse mit den Namen **TextmarkenSammeln**, **TextmarkenSprung** und **TextmarkenSortierung** jedes mal ablaufen werden, wenn die entsprechende Schaltfläche in unserem PopUp-Menü betätigt wird.

Etwas anders bei der Funktion, welche es erlaubt auf den Word-integrierten Dialog für Textmarken anzuzeigen. Diese Schaltfläche wurde mittels Code ganz einfach dupliziert. indem die entsprechende Befehls-ID beim *Add-Befehl* verwendet wird.

```
Private Const Leiste = "Standard"
Private Const Tag = "SW19991006"

Sub TextmarkenMenüEinrichten()
  CustomizationContext = NormalTemplate
```

```
  Dim Menü As CommandBarPopup
  Set Menü = CommandBars.FindControl(Type:=msoControlPopup, Tag:=Tag)
  If Not Menü Is Nothing Then Menü.Delete
  Set Menü = CommandBars(Leiste).Controls.Add(Type:=msoControlPopup)
  Dim Knopf As CommandBarButton
  With Menü
    .Tag = Tag
    .Caption = "&Textmarken"
    .BeginGroup = True
    .TooltipText = "Zeigt alle Textmarken im Dokument zur Auswahl an"
    .OnAction = "TextmarkenSammeln"
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .Tag = Tag & "1"
      .BeginGroup = True
      .Caption = "&Alphabetische Reihenfolge"
      .FaceId = 928
      .OnAction = "TextmarkenSortierung"
    End With
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .Tag = Tag & "2"
      .Caption = "Textmarken &Ein/Aus"
      .FaceId = 2636
      .OnAction = "TextmarkenToggle"
    End With
    Set Knopf = .Controls.Add(id:=758)
    With Knopf
      .BeginGroup = True
      .FaceId = 1546
    End With
    Set Knopf = .Controls.Add(Type:=msoControlButton)
    With Knopf
      .BeginGroup = True
      .Caption = "&Abbbrechen"
      .FaceId = 1019
    End With
  End With
  NormalTemplate.Save
End Sub

Private Sub TextmarkenSammeln()
  savSettings1 = NormalTemplate.Saved
  savSettings2 = ActiveDocument.AttachedTemplate.Saved
  Dim Menü As CommandBarPopup, Knopf As CommandBarButton
  Set Menü = CommandBars.ActionControl
  For Each c In Menü.Controls
    If c.Tag = Tag & "3" Or c.Tag = Tag & "4" Then c.Delete
  Next
  AnzTM = ActiveDocument.Bookmarks.Count
  If AnzTM = 0 Then
    Set Knopf = Menü.Controls.Add(Type:=msoControlButton, Before:=1)
    With Knopf
```

```
      .Tag = Tag & "4"
      .BeginGroup = True
      .Caption = "Keine Textmarken"
      .FaceId = 1786
    End With
  Else
    Dim oDoc As Document, x() As String, y() As String
    ReDim x(AnzTM - 1)
    ReDim y(AnzTM - 1, 1)
    Set oDoc = ActiveDocument
    For i = 0 To AnzTM - 1
      x(i) = oDoc.Bookmarks(i + 1).Name
      y(i, 1) = x(i)
      If oDoc.Bookmarks(i + 1).Range.StoryType = wdMainTextStory Then
        y(i, 0) = Format(oDoc.Bookmarks(i + 1).Range.Start, "00000000")
      Else
        y(i, 0) = 1000000000
      End If
    Next i
    If oDoc.Bookmarks.DefaultSorting = wdSortByLocation Then
      If AnzTM > 1 Then WordBasic.SortArray y()
    End If
    For i = 0 To AnzTM - 1
      Set Knopf = Menü.Controls.Add(Type:=msoControlButton, Before:=i + 1)
      With Knopf
        If oDoc.Bookmarks.DefaultSorting = wdSortByName Then
          .Caption = x(i)
        Else
        End If
        .Caption = y(i, 1)
        .Tag = Tag & "4"
        .FaceId = 1678
        .OnAction = "TextmarkenSprung"
      End With
    Next i
  End If
  Set Knopf = Menü.CommandBar.FindControl(Tag:=Tag & "1")
  If Not Knopf Is Nothing Then
    If ActiveDocument.Bookmarks.DefaultSorting = wdSortByName Then
      Knopf.State = msoButtonDown
    Else
      Knopf.State = msoButtonUp
    End If
  End If
  Set Knopf = Menü.CommandBar.FindControl(Tag:=Tag & "2")
  If Not Knopf Is Nothing Then
    If ActiveWindow.View.ShowBookmarks = True Then
      Knopf.State = msoButtonDown
    Else
      Knopf.State = msoButtonUp
    End If
  End If
  NormalTemplate.Saved = savSettings1
```

```
  ActiveDocument.AttachedTemplate.Saved = savSettings2
End Sub

Private Sub TextmarkenSprung()
  ActiveDocument.Bookmarks(CommandBars.ActionControl.Caption).Select
End Sub

Private Sub TextmarkenToggle()
  ActiveDocument.ActiveWindow.View.ShowBookmarks = _
     Not ActiveDocument.ActiveWindow.View.ShowBookmarks
End Sub

Private Sub TextmarkenSortierung()
  If CommandBars.ActionControl.State = msoButtonUp Then
    ActiveDocument.Bookmarks.DefaultSorting = wdSortByName
    CommandBars.ActionControl.State = msoButtonDown
  Else
    ActiveDocument.Bookmarks.DefaultSorting = wdSortByLocation
    CommandBars.ActionControl.State = msoButtonUp
  End If
End Sub
```

(Das praktische Textmarken-Utility)

Wichtige Eigenschaften und Methoden für Menüs

▲ Zurück zum Anfang

Eigenschaft oder Methode	Erklärung
Caption	Legt die **Bezeichnung** des Pop-Up Menüs fest
BeginGroup	Legt fest, ob vor dem Pop-Up Menü eine **Trennlinie** eingefügt werden soll
Visible	Legt fest, ob das Pop-Up Menü angezeigt wird
Enabled	Legt fest, ob das Pop-Up Menü angesprochen werden kann oder grau erscheint
CommanfBar	Gibt ein Untermenü eines Pop-Up Menüs zurück und zwar als CommandBar-Objekt. Verwenden Sie diese Eigenschaft um Zugriff auf ein Untermenü zu erlangen. Beispiel: Set Untermenü = CommandBars("Drawing").Controls(1).CommandBar.Controls(4)
Controls	Gibt ein Menüitem eines Pop-Up Menüs zurück. Verwenden Sie diese Eigenschaften um Zugriff auf ein Menüitem zu erlangen. Beispiel: Set Menüitem = CommandBars("Drawing").Controls(1)
TooltipText	Legt den Text fest, welcher erscheint, wenn der Benutzer einige Zeit mit dem Zeiger über dem Pop-Up Menü verharrt. Diese Information wird auch **Quick-Info** genannt und ist in der Regel zitronengelb hinterlegt.
OnAction	Hier geben Sie an, welcher Prozess (**Makro**) ausgeführt werden soll, wenn der Benutzer das Pop-Up Menü öffnet.
Tag	Hier können Sie dem Pop-Up Menü eine eindeutige Kennung verpassen, welche Sie z.B. zum Zeitpunkt auslesen können, wenn das Makro aufgerufen wird, welches Sie mit der Eigenschaft OnAction gesetzt haben. Die VBA-Methode FindControl erlaubt es auch, mittels dieser Kennung ein Pop-Up Menü über alle Symbolleisten zu finden. Für die Eindeutigkeit der Kennung (also des Tags) müssen Sie selbst besorgt sein. In der Praxis ist es besser, ein Pop-Up Menü über diese Eigenschaft zu suchen als über die Eigenschaft Caption, nicht zuletzt deshalb, weil die Tag-Eigenschaft nicht sprachspezifisch ist. Der Einsatz der FindControl-Methode wird später noch ausführlich behandelt.

Die FindControl-Methode

▲ Zurück zum Anfang

Die **FindControl-Methode** wurde in diesem Skript bereits an einigen Stellen erwähnt und eingesetzt. Sie dient dazu, den Standort einer Schaltfläche, von der wir nicht einmal sicher sind, ob es sie überhaupt gibt heraus zu finden. Ist die *FindControl-Methode* erfolgreich, dann wird die Schaltfläche als Objekt zurückgegeben und wir können auf die Schaltfläche zugreifen. Schlägt die Methode dagegen fehl, dann wird das Zielobjekt auf ***Nothing*** gesetzt. Diese Sache erweist sich in folgenden Fällen als nützlich:

- Wir möchten eine neue Schaltfläche hinzufügen, allerdings nur dann, wenn es diese noch nicht gibt.
- Wir möchten eine bestehend Schaltfläche mit einer neuen Version ersetzen und sollten nun wissen, ob die alte zuerst gelöscht werden soll.
- Wir möchten eine neue Schaltfläche relativ zu einer bestehenden Schaltfläche positionieren, also z.B. direkt vor einer bestimmten Schaltfläche oder direkt hinter dieser.

Man bedenke, dass jeder Benutzer Seine Symbolleisten mit oder ohne VBA-Kenntnisse jederzeit anpassen kann. Anzunehmen, dass das Menü *Datei* das erste Untermenü des Hauptmenüs in Word darstellt (und vieles andere), ist deshalb reine Spekulation.

Deshalb erweist sich weder das Ansprechen einer Schaltfläche über den Index (Reihenfolge auf einer Symbolleiste oder eines Menüs) noch das Ansprechen über die *Caption-Eigenschaft* (also über den Namen der Schaltfläche), .als besonders glücklich. (Leider) sieht man dies oft, vor allem in amerikanischen Programmen. Leider deshalb, weil die *Caption-Eigenschaft* sprachabhängig ist und die Bezeichnung oben drein noch vom Benutzer jederzeit geändert werden kann. Wie oft habe ich mich schon geärgert, wenn ich ein kleines Utility installiert habe und feststellen musste, dass sich die Schaltflächen bei jedem Installationsversuch kumulierten.

Die FindControl-Methode ermöglicht es uns über die folgenden Kriterien eine Schaltfläche aufzufinden:

- Mit der **ID-Eigenschaft** sind Sie gut bedient. Sie ist sprachenunabhängig und gewährt nur Lesezugriff. Damit kann sie natürlich auch nicht vom Benutzer geändert werden, auch nicht mittels VBA. Das ist gut um integrierte Befehle zu finden. Befehle, welche wir aber selbst erstellen, können keine ID haben, den diese Eigenschaft ist bekanntlich schreibgeschützt. Es bleibt uns aber einen guten Ausweg.
- Mit der **Tag-Eigenschaft** welche auch gesetzt werden kann sind wir für benutzerdefinierte Befehle gut bedient. Wir müssen nun nur noch sicher stellen, dass wir nicht den selben **Tag** mehrmals vergeben. Dazu legen wir uns am besten eine kleine Konvention zu recht. Verwenden Sie Ihre Initialen und hängen Sie das Datum im Format *JJJJMMTT* hinten dran. Sollten Sie an einem bestimmten Tag besonders fleissig sein und mehrere Befehle erstellen, hängen Sie anschliessend noch einen Korrelator an. Ihr Gedächtnis sollte frisch genug sein, um zu überblicken, was Sie an einem Tag machen. Da das Datum immer fortschreitet, besteht die Gefahr der Überlappung auch längerfristig nicht. Dieser *Tag* könnte konkret also wie folgt aussehen: *SW20000607*.

Hier nochmals die *FindControl-Methode* im Einsatz. Es wird nach der Schaltfläche mit der ID *11* gefahndet:

Deklarieren Sie das Objekt als **CommandBarControl** und nicht etwa spezifischer z.B. als *CommandBarButton*. Sie vermeiden damit einen Syntaxfehler, sollte die Schaltfläche von einem anderen Typus sein.

```
Sub SchaltflächeFindenRekursiv()
  Dim x As CommandBarControl
  Set x = CommandBars.FindControl(ID:=11)
  If x Is Nothing Then
    MsgBox "Dieses Schaltfläche wurde nicht gefunden."
  Else
    MsgBox "Die Bezeichnung dieser Schaltfläche lautet: " & _
       x.Caption & "und befindet sich auf der Symbolleiste " & _
          x.Parent.NameLocal & " an " & x.Index & " Stelle."
  End If
End Sub
```

(Mit der FindControl-Methode erhalten Sie Zugriff auf eine Schaltfläche deren ID oder Tag bekannt ist. Dabei brauchen Sie keine Kenntnisse darüber, auf welcher Symbolleiste sich diese Schaltfläche befindet. Die FindControl-Methode ermöglicht es auch, zu prüfen, ob eine bestimmte Schaltfläche überhaupt vorhanden ist.)

Die ActionControl-Methode

▲ Zurück zum Anfang

Die ActionControl-Methode wird dazu herangezogen um Zugriff auf jene Schaltfläche zu erhalten, welche die Prozedur aufgerufen hat, welche Sie bei der Eigenschaft *OnAction* gesetzt haben. Sie können dann auf einfache Weise diese Schaltfläche zum Beispiel verändern. Folgendes Beispiel sollen dies verdeutlichen.

Die erste Prozedur richtet einen Schaltknopf auf der Symbolleiste *Standard* ein und muss grundsätzlich nur einmal ablaufen. Die zweite Prozedur ist die *OnAction-Prozedur* und verändert das Aussehen des Schaltknopfes bei jedem Klick des Benutzers. Dabei muss die Prozedur nicht wissen, um welchen Schaltknopf es sich namentlich handelt, womit sie universell einsetzbar ist.

```
Sub KnopfEinrichten()
  Dim x As CommandBarButton
  Set x = CommandBars("Standard").Controls.Add(Type:=msoControlButton)
  x.Caption = "Flip"
  x.Style = msoButtonCaption
  x.OnAction = "FlipFlop"
End Sub
```

(Ein Schaltknopf für das nächste Experiment wird erstellt)

```
Sub FlipFlop()
  Dim Knopf As CommandBarButton
  Set Knopf = CommandBars.ActionControl
  If Knopf.State = msoButtonUp Then
    Knopf.Caption = "Flop"
    Knopf.State = msoButtonDown
    ActiveWindow.StyleAreaWidth = CentimetersToPoints(1.8)
  Else
    Knopf.Caption = "Flip"
    Knopf.State = msoButtonUp
    ActiveWindow.StyleAreaWidth = CentimetersToPoints(0)
  End If
End Sub
```

(Die ActionControl-Methode ermöglicht den Zugriff auf die Schaltfläche, welche soeben betätigt wurde, also jene Routine, welche in der Eigenschaft *OnAction* gesetzt wurde..)

Dank der ActionControl-Methode ist es auch möglich, generische *OnAction*-Routinen zu schreiben, also Prozesse, welche mehrere ähnlich veranlagte Schaltknöpfe unterstützen. Folgendes Beispiel, welches den Zugriff auf die gebräuchlichsten Sonderzeichen der französischen Sprache ermöglicht, soll dies verdeutlichen.

Die erste Prozedur richtet eine neue Symbolleiste ein und platziert darauf einige Schaltknöpfe. Diese Routine muss grundsätzlich nur einmal ablaufen. Die zweite Prozedur ist die *OnAction-Prozedur* und unterstützt sämtliche Schaltflächen auf dieser Symbolleiste.

```
Sub SymbolleisteFranzösischEinrichten()
  Dim Leiste As CommandBar
  Dim x As CommandBarButton
  CustomizationContext = NormalTemplate
  Set Leiste = CommandBars.Add(Name:="Französich", Position:=msoBarLeft)
  Leiste.Visible = True
  Set x = Leiste.Controls.Add(Type:=msoControlButton)
  x.Caption = "accent d' écu"
  x.OnAction = "FranzösischesZeichenEinfügen"
  x.Style = msoButtonCaption
  x.Parameter = "é"
  Set x = Leiste.Controls.Add(Type:=msoControlButton)
  x.Caption = "e grave"
  x.OnAction = "FranzösischesZeichenEinfügen"
  x.Style = msoButtonCaption
  x.Parameter = "è"
  Set x = Leiste.Controls.Add(Type:=msoControlButton)
  x.Caption = "c cédille"
  x.OnAction = "FranzösischesZeichenEinfügen"
  x.Style = msoButtonCaption
  x.Parameter = "ç"
End Sub
```

(Symbolleiste und Schaltknöpfe für das nächste Experiment werden erstellt.)

```
Sub FranzösischesZeichenEinfügen()
  Selection.TypeText Text:=CommandBars.ActionControl.Parameter
End Sub
```

(Die ActionControl-Methode ermöglicht auch den Einsatz von generischen *OnAction* -Routinen.)

Der CustomizationContext

▲ Zurück zum Anfang

Die Eigenschaft mit dem Namen *CustomizationContext* wurde hier schon mehrfach angesprochen und auch eingesetzt. Mit dieser Eigenschaft legen Sie fest, auf welcher Ebene eine Symbolleiste und deren Elemente erstellt, gelöscht bzw. verändert wird. Diese Eigenschaft entspricht der Einstellung, welche Sie im Word-Dialog ***Extras - Anpassen - Befehle - Speicher in:*** vornehmen können.

Es kommen sowohl geöffnete Dokumente wie auch geöffnete Dokumentenvorlagen in Frage, inklusive der globalen Vorlage mit dem Namen *Normal.dot*.

Zurück in VBA können Sie ein Dokument namentlich ansprechen (das gleiche gilt für Vorlagen) oder aber die speziellen Begriffe von Word-VBA benutzen. Hier einige Beispiele dazu:

- `CustomizationContext = NormalTemplate`
- `CustomizationContext = ActiveDocument`
- `CustomizationContext = ActiveDocument.AttachedTemplate`
- `CustomizationContext = Documents(1)`
- `CustomizationContext = Templates(1)`

Undokumentierte Typen von Schaltflächen

▲ Zurück zum Anfang

Wie schon Eingangs erwähnt, kennt MS Office eine recht grosse Fülle von Schaltflächen für Symbolleisten, welche sich sowohl durch ihr Aussehen wie auch durch ihre Funktion- und Einsatzweise unterscheiden. Allerdings können Sie ausschliesslich die besprochenen drei Arten selbst erstellen.

Ist dem wirklich so, oder haben wir da etwas verpasst? Es scheint so (dass haben Test ergeben), dass noch zwei zusätzliche Schaltflächen möglich wären, nämlich **Text-** und **DropDown**-Felder. Die Dokumentation dazu fehlt allerdings. In Anbetracht dessen und der Tatsache, dass diese beiden Schaltflächen irgendwie redundant sind (das Kombinationsfeld stellt ja eigentlich ein Textfeld kombiniert mit einem *DropDown-Feld* dar) verzichte ich auf eine ausführliche Beschreibung dieser zusätzlichen Elemente.

Nachfolgend ein kleines Beispiel für ein Textfeld. Obwohl die Eigenschaften dazu nicht dokumentiert sind Bedarf es nicht viel Phantasie eine solche Schaltfläche zu erstellen und zu steuern.

Die erste Prozedur erstellt die Schaltfläche und muss grundsätzlich nur einmal ablaufen. Die zweite Prozedur wird jedes Mal aufgerufen, wenn der Benutzer im Textfeld den Wert verändert und anschliessend die Eingabetaste drückt.

```
Sub TextSchaltfläche()
  Dim x As CommandBarControl
  Set x = CommandBars("Standard").Controls.Add(Type:=msoControlEdit)
  With x
    .Style = 1
    .Caption = "cm"
    .Width = CentimetersToPoints(3)
    .OnAction = "TextSchaltflächeExit"
    .Text = "1"
  End With
End Sub

Sub TextSchaltflächeExit()
  Dim x As CommandBarControl
  Set x = CommandBars.ActionControl
  If IsNumeric(x.Text) Then
    x.Caption = "Zoll"
    x.Text = x.Text * 2.54
  Else
  End If
End Sub
```

(Obwohl undokumentiert - auch Textfelder sind auf einer Symbolleiste möglich.)

Nachfolgend ein kleines Beispiel für ein *DropDown-Feld* (auch Listenfeld genannt). Der Unterschied von einem solchen Listenfeld zu einem Kombinationsfeld liegt darin, dass beim ersteren ausschliesslich ein Wert aus der Liste ausgewählt werden kann, wogegen das Kombinationsfeld es zusätzlich ermöglicht einen Wert direkt einzutippen. Obwohl die Eigenschaften dazu nicht dokumentiert sind Bedarf es nicht viel Phantasie eine solche Schaltfläche zu erstellen und zu steuern, sind diese doch weitgehend deckungsgleich mit den dokumentierten Eigenschaften eines Kombinationsfeldes.

Die erste Prozedur erstellt die Schaltfläche und muss grundsätzlich nur einmal ablaufen. Die zweite Prozedur wird jedes Mal aufgerufen, wenn der Benutzer im *DropDown-Feld* einen Eintrag auswählt.

```
Sub DropDownSchaltfläche()
  Dim x As CommandBarControl
  Set x = CommandBars("Standard").Controls.Add(Type:=msoControlDropdown)
  With x
    .Style = msoComboLabel
    .Caption = "Planeten"
    .Width = CentimetersToPoints(4)
    .AddItem "Merkur"
    .AddItem "Venus"
    .AddItem "Mars"
    .ListIndex = 1
    .OnAction = "DropDownSchaltflächeExit"
  End With
End Sub

Sub DropDownSchaltflächeExit()
  MsgBox "Das ist der " & CommandBars.ActionControl.ListIndex & _
     ". Planet in unserem Sonnensystem."
End Sub
```

(Obwohl undokumentiert - auch *DropDown-Felder* sind auf einer Symbolleiste möglich.)

Word-Interne Bezeichnung der Symbolleisten

▲ Zurück zum Anfang

Wie in diesem Skript schon erwähnt wurde, müssen Sie Symbolleisten mittels VBA mit dem Word-internen Namen ansprechen. Alternativ können Sie auch die ID der Symbolleiste heranziehen. Beim internen Namen handelt es sich in der Regel um die englische Bezeichnung der Symbolleiste. Die Nachfolgende Tabelle vermittelt die Gegenüberstellung von dem Namen, welcher Sie sich gewohnt sind (**Local Name** genannt) und dem internen Namen (**Name** genannt). Daneben gibt es auch noch die **ID** über welche eine Symbolleiste ebenfalls angesprochen werden kann.

Bei benutzerdefinierten Symbolleisten entsteht kein Problem, da in diesem Falle beide Namen identisch sind. Benutzerdefinierte Symbolleisten verfügen über keine ID. Die vorhandenen Symbolleisten können von Word-Version zu Word-Version leicht variieren, die nachfolgende Tabelle bezieht sich auf *Word2000*.

Interne Namen der Symbolleisten in Word

Interner Name	Local Name (Benutzername bzw. Name der deutschsprachigen Word-Version)	Typ	ID
3-D Settings	3D-Einstellungen	Normal	24
Font Paragraph	Absatzschriftart	PopUp	65
Curve Segment	Abschnitt krümmen	PopUp	80
ActiveX Control	ActiveX-Steuerelemente	PopUp	83
Refresh	Aktualisieren	Normal	37
Track Changes	Änderungen nachverfolgen	PopUp	70
Stop Recording	Aufzeichnung beenden	Normal	10
Align or Distribute	Ausrichten oder Verteilen	Normal	99
AutoShapes	AutoFormen	Normal	90
AutoSignature Popup	AutoSignaturmenü	PopUp	74
AutoText	AutoText	Normal	21
Field AutoText	AutoText-Feld	PopUp	75
AutoSummarize	AutoZusammenfassen	Normal	28
Block Arrows	Blockpfeile	Normal	93
Chinese Translation	Chinesische Übersetzung	PopUp	88
Database	Datenbank	Normal	4
Document Map	Dokumentstruktur	PopUp	76
Rotate or Flip	Drehen oder Kippen	Normal	100
Rotate Mode	Drehungsmodus	PopUp	85
Inline Picture	Eingebettete Grafik	PopUp	52
Send Mail	E-Mail senden	Normal	19
Endnotes	Endnoten	PopUp	41
Exit Design Mode	Entwurfsmodus beenden	Normal	29
Extended Formatting	Erweiterte Formatierung	Normal	20
Field Display List Numbers	Feldanzeige für Listenzahlen	PopUp	44
Fields	Felder	PopUp	42
Display Fields	Felder anzeigen	PopUp	43
Flowchart	Flussdiagramm	Normal	92
Formatting	Format	Normal	2
Shapes	Formen	PopUp	77
Forms	Formular	Normal	6
Form Fields	Formularfelder	PopUp	45
Frame Properties	Frameeigenschaften	PopUp	71
Frames	Frames	Normal	38
Fill Color	Füllfarbe	Normal	97
Function Key Display	Funktionstastenanzeige	Normal	34
Footnotes	Fußnoten	PopUp	46
Whole Table	Ganze Tabelle	PopUp	60

Interner Name	Local Name (Benutzername bzw. Name der deutschsprachigen Word-Version)	Typ	ID
Full Screen	Ganzer Bildschirm	Normal	7
Outlining	Gliederung	Normal	15
Picture	Grafik	Normal	26
Edit Picture	Grafik bearbeiten	Normal	8
Grammar	Grammatik	PopUp	67
Grammar (2)	Grammatik (2)	PopUp	68
Horizontal Line Popup	Horizontale Linie	PopUp	53
Hyperlink Context Menu	Hyperlink-Kontextmenü	PopUp	72
Hyperlink Menu	Hyperlinkmenü	PopUp	73
Drop Caps	Initial	PopUp	40
Inline ActiveX Control	Inline-ActiveX-Steuerelement	PopUp	89
Curve Node	Knoten krümmen	PopUp	79
Comment	Kommentar	PopUp	86
Header and Footer	Kopf- und Fußzeile	Normal	14
Headings	Kopfzeilen	PopUp	48
Curve	Krümmen	PopUp	78
Callouts	Legenden	Normal	91
Lines	Linien	Normal	95
Line Color	Linienfarbe	Normal	98
Lists	Listen	PopUp	51
Menu Bar	Menüleiste	MenuBar	36
Microsoft	Microsoft	Normal	13
Read Mail	Nachricht lesen	Normal	18
OLE Object	OLE-Objekt	PopUp	82
Outlook Read Mail	Outlook-E-Mail lesen	Normal	32
Outlook Send Mail	Outlook-E-Mail senden	Normal	33
Frames	Positionsrahmen	PopUp	47
Nudge	Präzisionsausrichtung	Normal	102
Borders	Rahmen	Normal	103
Spelling	Rechtschreibung	PopUp	66
Order	Reihenfolge	Normal	101
Shadow Settings	Schatteneinstellungen	Normal	25
Shading Color	Schattierungsfarbe	Normal	105
Font Color	Schriftfarbe	Normal	104
Print Preview	Seitenansicht	Normal	16
Mail Merge	Seriendruck	Normal	11
Script Anchor Popup	Skriptanchormenü	PopUp	50
Standard	Standard	Normal	1
Basic Shapes	Standardformen	Normal	96
Background Proofing Status Bar	Statusleiste für Rechtschreibprüfung im Hintergrund	PopUp	69
Stars & Banners	Sterne & Banner	Normal	94
Control Toolbox	Steuerelement-Toolbox	Normal	30
Tables	Tabellen	PopUp	54
Tables and Borders	Tabellen und Rahmen	Normal	3
Table Pictures	Tabellengrafiken	PopUp	58
Table Lists	Tabellenlisten	PopUp	57
Table Text	Tabellentext	PopUp	59
Table Headings	Tabellenüberschriften	PopUp	56
Table Cells	Tabellenzellen	PopUp	55
Text	Text	PopUp	62
Text Box	Textfeld	Normal	31
Text Wrapping	Textfluss	Normal	107
Reviewing	Überarbeiten	Normal	27

Interner Name	Local Name (Benutzername bzw. Name der deutschsprachigen Word-Version)	Typ	ID
Track Changes Indicator	Überarbeitungsanzeige	PopUp	87
Font Popup	Untermenü Schrift	PopUp	64
Floating Picture	Unverankerte Grafik	PopUp	81
Linked Table	Verknüpfte Tabelle	PopUp	61
Linked Headings	Verknüpfte Überschriften	PopUp	49
Linked Text	Verknüpfter Text	PopUp	63
Visual Basic	Visual Basic	Normal	9
Web	Web	Normal	22
Web Tools	Webtools	Normal	35
Word for Windows 2.0	Word für Windows 2.0	Normal	17
WordArt	WordArt	Normal	23
WordArt Context Menu	WordArt-Kontextmenü	PopUp	84
Drawing	Zeichnen	Normal	5
Cell Alignment	Zellausrichtung	Normal	106
Master Document	Zentraldokument	Normal	12
Clipboard	Zwischenablage	Normal	108

Symbolleisten schützen

▲ Zurück zum Anfang

Symbolleisten können geschützt werden und zwar in einer Weise, dass für den Benutzer keine oder nur noch bestimmte Änderungen zugelassen sind. MS Office stellt hier sogar einen sehr differenzierten Schutz zur Verfügung.

Die Eigenschaft dazu heisst **Protection**. Das einzige, was Sie beachten müssen, kumulieren bzw. addieren Sie die Werte. Wenn sie dies missachten kommt immer nur der zuletzt gesetzte Wert zum Zuge.

Wenn Sie also z.B. verhindern möchten, dass der Benutzer die Symbolleiste ausblenden kann **und** auch nicht anpassen kann dann verwenden Sie folgenden Code:

```
Sub SymbolleisteSchutzfaktor()
  CustomizationContext = NormalTemplate
  CommandBars("Formatting").Protection = _
     msoBarNoChangeVisible + msoBarNoCustomize
End Sub
```

(Symbolleisten können vor Veränderungen sehr differenziert geschützt werden. Die Werte für die Eigenschaft *Protection* müssen dabei addiert werden.)

Etwas kürzer wird der Code, wenn Sie die Summe der Werte explizit einsetzen. Folgender Code bewirkt also das selbe, wie der oben abgedruckte Code:`CommandBars("Formatting").Protection = 9`. Der numerische Wert um das Ausblenden zu verhindern lautet nämlich *8* und der Wert um das Anpassen zu verhindern laute *1*. Dies ergibt in der Summe *9*. Diese Addition liefert immer einen eindeutigen Wert, da der Teilwert für eine bestimmte Eigenschaft immer eine 2-er Potenz darstellt.

Um den Schutz einer Symbolleiste wieder ganz aufzuheben, vergeben Sie den Wert *0*.

Im folgenden eine Tabelle, welche es Ihnen erlaubt, diese Zahl direkt herauszulesen:

Schutzfaktor für Symbolleisten

Summe	Vertikaler Dock erlaubt	Abdocken erlaubt	Ein-/ Ausblenden	Verschieben erlaubt	Grösse veränderbar	Anpassen erlaubt
Office-Konstante	msoBar NoVerticalDock	msoBar NoChangeDock	msoBar NoChangeVisible	MsoBar NoMove	MsoBar NoResize	msoBar NoCustomize
Eigenwert	**32**	**16**	**8**	**4**	**2**	**1**
0	X	X	X	X	X	X
1	X	X	X	X	X	O
2	X	X	X	X	O	X
3	X	X	X	X	O	O
4	X	X	X	O	X	X
5	X	X	X	O	X	O
6	X	X	X	O	O	X
7	X	X	X	O	O	O
8	X	X	O	X	X	X
9	X	X	O	X	X	O
10	X	X	O	X	O	X
11	X	X	O	X	O	O
12	X	X	O	O	X	X
13	X	X	O	O	X	O
14	X	X	O	O	O	X
15	X	X	O	O	O	O
16	X	O	X	X	X	X
17	X	O	X	X	X	O
18	X	O	X	X	O	X
19	X	O	X	X	O	O
20	X	O	X	O	X	X
21	X	O	X	O	X	O
22	X	O	X	O	O	X
23	X	O	X	O	O	O
24	X	O	O	X	X	X
25	X	O	O	X	X	O
26	X	O	O	X	O	X
27	X	O	O	X	O	O
28	X	O	O	O	X	X
29	X	O	O	O	X	O
30	X	O	O	O	O	X
31	X	O	O	O	O	O
32	O	X	X	X	X	X
33	O	X	X	X	X	O
34	O	X	X	X	O	X
35	O	X	X	X	O	O
36	O	X	X	O	X	X
37	O	X	X	O	X	O
38	O	X	X	O	O	X
39	O	X	X	O	O	O
40	O	X	O	X	X	X
41	O	X	O	X	X	O
42	O	X	O	X	O	X
43	O	X	O	X	O	O
44	O	X	O	O	X	X
45	O	X	O	O	X	O
46	O	X	O	O	O	X
47	O	X	O	O	O	O

Summe	Vertikaler Dock erlaubt	Abdocken erlaubt	Ein-/ Ausblenden	Verschieben erlaubt	Grösse veränderbar	Anpassen erlaubt
Office-Konstante	msoBar NoVerticalDock	msoBar NoChangeDock	msoBar NoChangeVisible	MsoBar NoMove	MsoBar NoResize	msoBar NoCustomize
Eigenwert	**32**	**16**	**8**	**4**	**2**	**1**
48	O	O	X	X	X	X
49	O	O	X	X	X	O
50	O	O	X	X	O	X
51	O	O	X	X	O	O
52	O	O	X	O	X	X
53	O	O	X	O	X	O
54	O	O	X	O	O	X
55	O	O	X	O	O	O
56	O	O	O	X	X	X
57	O	O	O	X	X	O
58	O	O	O	X	O	X
59	O	O	O	X	O	O
60	O	O	O	O	X	X
61	O	O	O	O	X	O
62	O	O	O	O	O	X
63	O	O	O	O	O	O

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen?

Microsoft

► Menus and Toolbars

► PRIME Bookmark Popup for Microsoft Word 97/2000

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/OnAction.htm

René Probst, Juli 2000

Änderungen

- --- nil ---

Wie kann ich alle VBA-Module (Makros) einer Vorlage oder eines Dokuments drucken?

◄ Zurück zum Hauptindex

Problemstellung

> **Aus dem VBA-Editor kann ich zwar einzelne Module (Makros) drucken. Allerdings ist der linke Rand beim Ausdruck viel zu schmal. Ich habe auch keine Möglichkeit gefunden, alle Makros einer Vorlage oder eines Dokumentes miteinander zu drucken.**

Lösung

Folgender Code fordert den Benutzer als erstes auf, eine Dokumentenvorlage oder ein Dokument zu öffnen, und druckt dann alle **Module** (Makros, **UserForms**, **Klassenmodule** und der Code welcher allenfalls unter *(**ThisDocument***) existiert aus.

VBA-Code

```
' Diese Routine druckt alle Module einer Dokumentenvorlage oder eines Dokumentes.
' Als erstes werden Sie aufgefordert, eine Vorlage oder ein Dokument zu öffnen.
'
' Erstellt am 02/09/99 von René Probst / eku / CH-8152 Glattbrugg
'
Private Const asTitle = "VBA-Module drucken"
Private Assi As Boolean
Private asVisible As Boolean
Private oBlase As Balloon

Sub ModuleDrucken()
  Set oAssistant = Assistant
  If Not oAssistant Is Nothing Then
    Assi = True
    asVisible = Assistant.Visible
    Assistant.Animation = msoAnimationGreeting
    Antwort = BlaseAnzeigen
  Else
    Assi = False
  End If
  Set oAssistant = Nothing
  If Antwort = -2 Then HappyEnd
  Dim oDoc As Document
  Dim rDoc As Document
  With Dialogs(wdDialogFileOpen)
    .Name = Options.DefaultFilePath(wdUserTemplatesPath) & "\"
    Antwort = .Show
  End With
  If Antwort = 0 Then HappyEnd
  Set oDoc = ActiveDocument
  Set rDoc = Documents.Add
  System.Cursor = wdCursorWait
  Application.ScreenUpdating = False
  Seiteeinrichten
  With Selection
    .LanguageID = wdNoProofing
```

```
    .Style = wdStyleHeading1
    .TypeText Text:="Module im Projekt " & oDoc.VBProject.Name & _
       " (" & oDoc.Name & ")" & vbCrLf
    Anz = 0
    For Each x In oDoc.VBProject.VBComponents
      If x.CodeModule.CountOfLines > 0 Then
        Anz = Anz + 1
        .Style = wdStyleHeading2
        .TypeText Text:=x.Name & " (" & EditType(x.Type) & ")" & vbCrLf
        RahmenHinzufügen 9, 9
        .Style = wdStyleMacroText
        .TypeText Text:=x.CodeModule.lines(1, x.CodeModule.CountOfLines) & vbCrLf
        RahmenHinzufügen 6, 45
      End If
    Next
    .HomeKey Unit:=wdStory
    If Anz = 0 Then
      .Style = wdStyleNormal
      .Font.Italic = True
    End If
  End With
  oDoc.Saved = True
  oDoc.Close
  rDoc.PrintOut
  Assistant.Animation = msoAnimationCharacterSuccessMajor
  HappyEnd
End Sub

Private Sub Seiteeinrichten()
  With ActiveDocument.PageSetup
    .TopMargin = CentimetersToPoints(1.2)
    .BottomMargin = CentimetersToPoints(1.8)
    .LeftMargin = CentimetersToPoints(2.5)
    .RightMargin = CentimetersToPoints(1.2)
  End With
  ActiveWindow.ActivePane.View.SeekView = wdSeekCurrentPageFooter
  With Selection
    .ParagraphFormat.Alignment = wdAlignParagraphRight
    x = .Information(wdHorizontalPositionRelativeToTextBoundary)
    .ParagraphFormat.Alignment = wdAlignParagraphLeft
    .ParagraphFormat.TabStops.ClearAll
    .ParagraphFormat.TabStops.Add Position:=x, Alignment:=wdAlignTabRight
    .TypeText Text:="Stand: " & Format(Date, "d.mmm.yyyy") & vbTab
    .Fields.Add Range:=.Range, Type:=wdFieldPage
    .EndKey
    .TypeText Text:=" von "
    .EndKey
    .Fields.Add Range:=.Range, Type:=wdFieldNumPages
  End With
  ActiveWindow.ActivePane.View.SeekView = wdSeekMainDocument
End Sub

Private Sub RahmenHinzufügen(tvor, tnach)
  With Selection
    .Style = wdStyleNormal
    .Font.Size = 1
    With .ParagraphFormat
      .Space1
```

```
      .SpaceBefore = tvor
      .SpaceAfter = tnach
      With .Borders(wdBorderTop)
        .LineStyle = wdLineStyleSingle
        .LineWidth = wdLineWidth225pt
        .Color = wdColorGray75
      End With
    End With
    .TypeParagraph
  End With
End Sub

Private Function EditType(tType)
  Select Case tType
    Case 1: t = "Modul"
    Case 2: t = "Klassenmodul"
    Case 3: t = "UserForm"
    Case 100: t = "Objekt"
  End Select
  EditType = t
End Function

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = asTitle
    tmp = "Diese Routine druckt alle Module einer Dokumentenvorlage oder eines "
    tmp = tmp & "Dokumentes." & vbCrLf & vbCrLf & "Als erstes werden Sie "
    .Text = tmp & "aufgefordert, eine Vorlage oder ein Dokument zu öffnen."
    .Button = msoButtonSetOkCancel
    BlaseAnzeigen = .Show
  End With
End Function

Private Sub HappyEnd()
  Application.ScreenUpdating = True
  System.Cursor = wdCursorNormal
  If Assi = True Then Assistant.Visible = asVisible
  End
End Sub
```

(Druckt alle Module einer Dokumentvorlage oder eines Dokumentes aus.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/VBAMd.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Welche Schalter (Switches) gibt es um die Word-Anwendung zu starten?

◄ Zurück zum Hauptindex

Frage

> Die meisten Anwendungen kennen nützliche Schalter, welche es erlauben, die Anwendung zu ganz bestimmten Zwecken zu starten. Welcher Schalter kennt Word?

Antwort

Hier eine Auflistung mit Erklärungen für die Schalter in **Word 97**. Für andere Word-Versionen, besuchen Sie bitte den eingeblendeten Link am Ende des Kapitels.

Tabelle der Schalter beim Starten der Word-Anwendung

Syntax	Bedeutung	Bemerkung
`winword.exe /q`	**Word2000** startet ohne ***splash screen***	**Nur Word2000 Nach dem Einspielen des Office2000 Service Release 1!**
`winword.exe <Dateiname>`	Word wird mit der nominierten Datei geöffnet.	Lange Dateinamen oder Pfadnamen müssen in Gänsefüsschen eingebettet werden.
`winword.exe <Dateiname1> <Dateiname2> ... <DateinameX>`	Word wird mit mehreren nominierten Dateien geöffnet.	Lange Dateinamen oder Pfadnamen müssen in Gänsefüsschen eingebettet werden.
`winword.exe /n`	Word startet ohne eine Dokument zu generieren.	---
`winword.exe /a`	Word startet ohne (globale) Vorlagen oder Add-Ins.	Auch die Standardvorlage Normal.dot wird nicht geladen. Dieser Schalter eigenen sich sehr gut um Probleme zu finden bzw. zu beheben.
`winword.exe /t<Vorlage>`	Word startet mit der nominierten Vorlage.	Zwischen dem Schalter "/t" und dem Name der Vorlage darf kein Leerschlag sein. Lange Dateinamen oder Pfadnamen müssen in Gänsefüsschen eingebettet werden. Kann leider nicht mit dem Schalter /m kombiniert werden.
`winword.exe /m<Makroname>`	Word startet und ruft den nominierten Prozess (Makro) auf.	Zwischen dem Schalter "/m" und dem Name der Vorlage darf kein Leerschlag sein. Kann leider nicht mit dem Schalter /t kombiniert werden.
`winword.exe /m`	Word startet ohne die Markos mit den reservierten Namen AutoExec, AutoNew oder AutoOpen aufzurufen.	---
`winword.exe /r`	Word registriert sich selbst.	Word 97 startet nicht sichtbar und gibt auch keine Endmeldung aus. Sie sollten aber deutlich wahrnehmen können, wie die Festplatte für 2-3 Sekunden arbeitet. In **Word2000** erfolgt eine Endmeldung. Dieser Schalter eigenen sich sehr gut um Probleme zu finden bzw. zu beheben.

Anmerkung

- Ist der Schalter, welcher genau das macht, nachdem Sie suchen nicht in dieser Liste enthalten? - Dann haben Sie wahrscheinlich Pech gehabt. Nach meinem Informationsstand kennt Word keine "versteckten Schalter", also Schalter welche nicht im folgenden Artikel von Microsoft dokumentiert wären.

Links zu diesem Thema

Microsoft

Englisch

► **WD: Word for Windows Startup Switches**

► **WD2000: Word Startup (Command-Line) Switches and Their Purpose**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Schalter.htm

René Probst, Oktober 1999

Änderungen

- 23. September 2000 - Schalter zur Unterdrückung des splash screens

Wie kann ich programmtechnisch ex MS Word auf andere Office-Anwendungen zugreifen?

Wie kann ich aus anderen Anwendungen auf MS Word zugreifen?

◄ Zurück zum Hauptindex

Diese Seite

▼ Excel mit einer leeren Arbeitsmappe starten
▼ Excel mit einer bestimmten Vorlage (XLT) starten
▼ Bestehende Excel-Arbeitsmappe öffnen
▼ Ganzes Excel-Tabellenblatt in ein Array (Datenfeld) laden
▼ MS Word aus Excel mit einem leeren Dokument starten
▼ Bestehendes Word-Dokument aus Excel öffnen
▼ Verweise auf Objektbibliotheken setzen
▼ Automatisieren - aber richtig!
▼ Von MS Word auf Access zugreifen
▼ Mit «DAO» auf Access zugreifen
▼ Mit «ADO» auf Access zugreifen
▼ Aus Word ein Makro in Access starten
▼ Mit «DAO» Daten aus einem Excel-Tabellenblatt ziehen, ohne die Arbeitsmappe zu öffnen
▼ Von MS Word auf MS Outlook2000 zugreifen
▼ MS Word Seriendruck aus dem Internet Explorer automatisiert ausführen
▼ Links zu diesem Thema

Vorausbemerkungen

▲ Zurück zum Anfang

> Flexibel und zukunftssicher fahren Sie mit *Office Automation* (*CreateObject-* und *GetObject*-Methode), früher auch *OLE-Automation* genannt. Diese Methoden erlauben es eine Office-Anwendung zu starten und ermöglicht gleichzeitig den Zugriff auf das Objekt-Modell dieser Server-Anwendung.
> Um auf eine Access-Datenbank aus einer anderen Anwendung zuzugreifen können Sie entweder mit *DAO* (*Data Access Object* oder *ADO* (*ActiveX Data Object* arbeiten.

Excel mit einer leeren Arbeitsmappe starten

▲ Zurück zum Anfang

Der folgende Code startet Excel mit einer leeren Arbeitsmappe. Kopieren Sie den Code in ein neues Modul und testen Sie ihn dann aus.

```
Sub ExcelMitLeeremTabellenblattStarten()
  Dim xlAnw As Object
  Set xlAnw = CreateObject("Excel.Application")
  xlAnw.Visible = True
  xlAnw.WindowState = -4137 '-4143 = Normal, -4137 = Maximized; -4140 = Minimized
  xlAnw.Workbooks.Add
  xlAnw.Activate
  'xlWkb.Parent.Quit
  Set xlAnw = Nothing
End Sub
```

(Excel wird mit einer leeren Arbeitsmappe gestartet...)

Excel mit einer bestimmten Vorlage (XLT) starten

▲ Zurück zum Anfang

Der folgende Code startet Excel mit einer bestimmten Vorlage /XLT). Passen Sie Pfad und Name der Vorlage im Kopf des Codes an.

```
Private Const Vorlage = "C:\Vorlagen\Tabellenvorlagen\Reisekosten-Abrechnung.xlt"

Sub ExcelMitVorlageStarten()
  Dim xlAnw As Object
  Set xlAnw = CreateObject("Excel.Application")
  xlAnw.Visible = True
  xlAnw.WindowState = -4137 '-4143 = Normal, -4137 = Maximized; -4140 = Minimized
  xlAnw.Workbooks.Add Template:=Vorlage
  'xlWkb.Parent.Quit
  Set xlAnw = Nothing
End Sub
```

(Excel wird mit einer einer bestimmten Vorlage gestartet...)

Bestehende Excel-Arbeitsmappe öffnen

▲ Zurück zum Anfang

Der folgende Code öffnet eine vom Ihnen nominierte Excel Mappe und funktioniert auch dann fehlerfrei, wenn die Arbeitsmappe bereits geöffnet sein sollte. Sollte die Mappe bereits geöffnet sein, ist der Vorgang sehr schnell. In diesem Falle muss nur die Verbindung von Word nach Excel hergestellt werden.

Kopieren Sie den Code in ein neues Modul, passen Sie Pfad und Dateiname im Kopf des Codes an und testen Sie ihn dann aus.

```
Private Const Pfad = "C:\Eigene Dateien\TestData.xls"
Sub ExcelMitBestehenderMappeStarten()
  Dim xlAnw As Object
  Dim xlMappe As Object
  Set xlMappe = GetObject(Pfad)
  Set xlAnw = xlMappe.Parent
  xlAnw.WindowState = -4137 '-4143 = Normal, -4137 = Maximized; -4140 = Minimized
  xlAnw.Visible = True
  xlAnw.Windows(1).Visible = True
  xlAnw.ActiveWindow.WindowState = -4137
  xlMappe.Saved = True
  'xlAnw.Parent.Quit
  Set xlAnw = Nothing
  Set xlMappe = Nothing
End Sub
```

(...die Excel-Mappe mit dem nominierten Namen wird geöffnet, falls diese schon geöffnet sein sollte, wird eine Verbindung vom Programm zur Mappe erstellt.)

Ganzes Excel-Tabellenblatt in ein *Array* (Datenfeld) laden

▲ Zurück zum Anfang

Einleitung

Die meisten Exceltabellen (Ausnahmen sind Pivot-Tabellen, etc.) lassen sich mit sehr wenig Code gänzlich in einem zweidimensionalen *Array* (auch Datenfeld genannt) abbilden. Wenn Sie das tun können Sie alsbald die Excel-Mappe schliessen und Word-intern mit den Daten von Excel weiter arbeiten. Der Zeitgewinn bei der Verarbeitung ist bei Rechnern, welche über wenig physischen Arbeitsspeicher (RAM) verfügen schon hoch, bei einem üppig ausgestatteten Rechner, welcher nur selten Daten in den Arbeitsspeicher auslagern muss, phänomenal.

Der nachfolgende VBA-Code bildet ein ganzes Excel-Tabellenblatt in einem solchen Array mit dem Namen x ab. Sie können dann z.B. die Zelle B7 unter der Variablen x(7, 2) ansprechen.

Passen Sie im Kopf des Codes sowohl Pfad und Name der Arbeitsmappe wie auch den Namen des Tabellenblattes an. Als Test, ob der Array erfolgreich geladen wurde, erhalten Sie in einer Meldungsbox den Inhalt der Zelle A1 zurück.

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\TestData.xls"
Private Const Blatt = "Tabelle1"

Sub ExcelSheetToArray()
  Dim xlMappe As Object
  Set xlMappe = GetObject(Pfad)
  MaxRows = xlMappe.Worksheets(Blatt).Cells.SpecialCells(11).Row
  MaxCols = xlMappe.Worksheets(Blatt).Cells.SpecialCells(11).Column
  ReDim x(MaxRows, MaxCols)
  For RowIndex = 1 To MaxRows
    For ColIndex = 1 To MaxCols
      x(RowIndex, ColIndex) = xlMappe.Worksheets(Blatt).Cells(RowIndex, ColIndex).Value
    Next ColIndex
  Next RowIndex
  xlMappe.Parent.Quit
  Set xlMappe = Nothing
  MsgBox "Der Inhalt der Zelle A1 lautet: " & vbCrLf & x(1, 1)
End Sub
```

(Viele Excel-Tabellenblätter sind von einer Beschaffenheit, dass sie sich mit Leichtigkeit als zweidimensionalen *Array* abbilden lassen.)

MS Word aus Excel mit einem leeren Dokument starten

▲ Zurück zum Anfang

Der folgende Code startet Word aus Excel mit einem leeren Dokument. Kopieren Sie den Code in Excel in ein neues Modul und testen Sie ihn dann aus.

```
Sub WordMitLeeremDokumentStarten()
  Dim wdAnw As Object
  Set wdAnw = CreateObject("Word.Application")
  wdAnw.Visible = True
  wdAnw.Documents.Add
  wdAnw.WindowState = 1 '0 = Normal; 1 = Maximized; 2 = Minimized
  wdAnw.Activate
  'wdAnw.Quit
  Set wdAnw = Nothing
End Sub
```

(Word wird aus Excel mit einem leeren Dokument gestartet...)

Bestehendes Word-Dokument aus Excel öffnen

▲ Zurück zum Anfang

Der folgende Code öffnet ein vom Ihnen nominiertes Word-Dokument und funktioniert auch dann fehlerfrei, sollte das Dokument bereits geöffnet sein. Ist das Dokument bereits geöffnet, geht der Vorgang sehr schnell. In diesem Falle muss nur die Verbindung von Excel zum Word-Dokument hergestellt werden. Wenn nicht, dauert der Vorgang eine Weile.

Kopieren Sie den Code in ein neues Modul, passen Sie Pfad und Dateiname des zu öffnenden Worddokumentes im Kopf des Codes an und testen Sie ihn dann aus.

```
Private Const Pfad = "C:\Eigene Dateien\Test.doc"

Sub WordMitBestehendemDokumentStarten1()
  Dim wdAnw As Object
  Dim wdDok As Object
  Set wdDok = GetObject(Pfad)
  Set wdAnw = wdDok.Parent
  wdAnw.Visible = True
  wdAnw.WindowState = 1 '0 = Normal; 1 = Maximized; 2 = Minimized
  wdAnw.Activate
  'wdDok.Close
  'wdAnw.Quit
  Set wdAnw = Nothing
  Set wdDok = Nothing
End Sub
```

(Ein bestehendes Word-Dokument wird aus Excel geöffnet.)

Hier noch ein Code-Beispiel, welches einen etwas anderen Weg verfolgt und auch Fehlerkonditionen abfangen kann. Im Gegensatz zum vorherigen Beispiel wird das Word-Dokument beim Öffnen hier mit den neusten Daten von Excel aktualisiert. Bedingung dafür ist es, dass in Word die Option *Bearbeiten - Verknüpfungen... - Aktualisieren - Automatisch* aktiviert ist. Überprüfen Sie zusätzlich die Option *Extras - Optionen... - Allgemein - Automatische Verknüpfungen beim Öffnen aktualisieren*.

```
Private Const Pfad = "C:\Eigene Dateien\Test.doc"
Private wdAnw As Object
Private wdDok As Object

Sub WordMitBestehendemDokumentStarten2()
  On Error Resume Next
  Set wdDok = GetObject(, "Word.Application") 'Word Instanz suchen
  Fehler = Err.Number
  On Error GoTo 0
  If Fehler = 429 Then
    Set wdAnw = CreateObject("Word.Application") 'Word Instanz generieren
  Else
    Set wdAnw = GetObject(, "Word.Application") 'Word Instanz verbinden
  End If
  wdAnw.Visible = True 'Instanz sichtbar machen
  wdAnw.WindowState = 1 'Fenster maximieren
  Flag = False
  For Each Dok In wdAnw.Documents 'Mein Dokument suchen
    If Dok.FullName = Pfad Then
      Flag = True
      Set wdDok = Dok 'Verbindung mit Dokument erstellen
      Exit For
    End If
  Next
  If Flag = False Then
    On Error Resume Next
    Set wdDok = wdAnw.Documents.Open(Filename:=Pfad) 'Dokument öffnen
    If Err.Number > 0 Then 'Wenn Dok nicht vorhanden
      MsgBox Err.Description
      HappyEnd
    End If
    On Error GoTo 0
  End If
  'Ihr Code
  'Ihr Code
  'Ihr Code
  HappyEnd
End Sub

Private Sub HappyEnd()
  Set wdDok = Nothing 'tidy up
  Set wdAnw = Nothing
End Sub
```

(Eine Alternative zum obigen Beispiel, welche auch auf Fehlerkonditionen eingeht.)

Verweise auf Objektbibliotheken setzen

▲ Zurück zum Anfang

Sie mögen sich vielleicht gewundert haben, wieso die Code-Beispiele hier wenig aussagende absolute numerische Werte satt wie üblich Word- und Excel-Konstanten einsetzen. So wird z.B.

```
xlAnw.WindowState = -4137
```

verwendet, satt

```
xlAnw.WindowState = xlMaximized.
```

Der Grund ist folgender: Die Client-Applikation, also die aufrufende Anwendung, kennt diese Konstanten gar nicht. Somit würde ein leerer Wert an die Server-Anwendung geschickt was natürlich einen Syntaxfehler hervor rufen würde. Dieses Verhalten können Sie aber ändern und es ist auch sicher ratsam, dies zu tun. Richten Sie ganz einfach in der Client-Anwendung einen Verweis auf die Objektbibliothek der Server-Anwendung ein, welche Sie erreichen möchten. Um von Word einen Objektverweise auf die Excel-Objektbibliothek einzurichten, gehen Sie wie folg vor:

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** über ***Datei - Öffnen*.**
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
3. Wählen Sie dort vom Hauptmenü ***Extras - Verweise...***.
4. Suchen Sie in der Liste nach der Objektbibliothek mit der Bezeichnung **Microsoft Excel Object Library 8.0 Word 97** bzw. **Microsoft Excel Object Library 9.0 Word2000** und setzen Sie dort das Häkchen.
5. Speichern Sie die Normal.dot. Word sollte sich diese Einstellung nun für die Zukunft merken.

Vorteile von Verweisen

Wenn Sie einen Verweis auf die Objektbibliothek der Server-Anwendung einrichten handeln Sie sich Vorteile ein, welche sich sowohl während der Entwurfzeit, wie auch während der Laufzeit positiv bemerkbar machen:

- Sie können Objekte nun exakt deklarieren, d.h. statt `Dim xlMappe As Object` steht Ihnen nun folgende Syntax zur Verfügung: `Dim xlMappe As Excel.Workbook`.
- Alle bekannten anwendungsintegrierten **Konstanten** der Server-Anwendung stehen ihnen nun auch in der Client-Anwendung zur Verfügung.
- Beim Entwickeln des Codes, können Sie nun vom ***Quick-Info-Feature*** und von der ***AutoVervollständigung*** des VBA-Editors profitieren, da Objekte, Eigenschaften und Methoden der Server-Anwendung nun auch in der Client-Anwendung bekannt sind.
- Etwaige Fehlermeldungen, welche Sie zurück erhalten sind nun in der Regel nun aussagekräftiger.
- Die Ausführung des Code gewinnt an Performanz.

Automatisieren - aber richtig!

▲ Zurück zum Anfang

Einleitung

Office Automation kommt sehr oft dann zum Zuge, wenn Abläufe automatisiert werden sollen. Dass eine Automatisation so ihre Tücken hat, erfährt man spätestens dann, wenn man sich konkret damit befasst. Der intelligente und flexible Mensch fällt nun aus und ohne Vorkehrungen wird kein Stück Code eine Frage, wie z.B. die folgende beantworten:
Die Seitenränder liegen ausserhalb des druckbaren Bereiches. - Wollen Sie weiter fahren?

Nun wartet also die Anwendung bzw. unser Code auf *Godo*. Der Code wird nicht weiter abgebaut, das Ziel der Aufgabe, obwohl der Code eigentlich fehlerfrei daher kommt, ist verfehlt. Diese Geschwätzigkeit z.B. von Word steht einer Vollautomatisation also im Wege. Richtig automatisieren bedeutet deshalb vor allem eines: **Vorsehen und Vorbeugen**!

Im folgenden Beispiel werden einige Techniken und Vorkehrungen vorgezeigt. Diese Mechanismen sollen verhindern, dass ein Prozess wegen einer vielleicht banalen Angelegenheit frühzeitig stoppt.

Der Code verfügt über folgende einfache Funktionalität:

- Die Word-Anwendung wird als Server-Anwendung von einem Klienten gestartet.
- Ein nominiertes Dokument wird geöffnet.
- Dieses Dokument wird gedruckt.
- Die Anwendung wird sauber und automatisch geschlossen.

Techniken

Damit das Vorhaben möglichst störungs- und interventionsfrei Abläuft werden folgende Techniken angewandt:

- Das Anwendungsfenster wird versteckt (`Visible = False`). Das braucht weniger Ressourcen und irritiert den Benutzer nicht.
- Allfällige Meldungsboxen sollen nicht angezeigt werden. (`DisplayAlerts = 0`). Da die meisten dieser Dialog modaler Natur sind, also zwingend vom Benutzer eine Intervention verlangen bevor der Code weiter läuft, ist diese Sache äusserst wichtig.
- Das Dokument wird schreibgeschützt geöffnet, denn es könnte ja breites in Bearbeitung (*in use*, also allokiert) sein. (`ReadOnly:=True`)
- Falls das Dokument eine Konvertierung erfordert (z.B. Textdatei), dann regelt Word dies automatisch, also ohne Rückfrage beim Benutzer. (`ConfirmConversions:=False`). Diese Parametrisierung überspielt die Einstellung im Word-Dialog *Extras - Optionen*.
- Das Dokument wird nicht in die Liste der *Zuletzt geöffneten Dokumente* übernommen. Diese Vorkehrung ist nicht vital, aber irgendwie folgerichtig. Sie verhindert, dass der Benutzer irritiert wird, wenn er in den Word-Dialog *Datei* wechselt. (`AddToRecentFiles:=False`)
- Es wird verhindert, dass das Dokument im Hintergrund gedruckt wird. Dies ist äusserst vital, würde Word doch sonst enden, bevor das Dokument fertig gedruckt ist. Der Druck würde also an zufälliger Stelle abgewürgt. (`Options.PrintBackground = False`). Statt diese Option umzupolen, welche der Benutzer im Word-Dialog *Extras - Optionen* vorfindet, könnt man auch codieren `oDoc.PrintOut Background:=False`. Aber Achtung: diese Variante arbeitet in Kombination mit *DisplayAlert* nicht korrekt. Dieses Fehlverhalten ist bei Microsoft dokumentiert. Benutzen Sie also besser die Variante mit der Option, wie dies auch im Beispiel realisiert ist.
- Der benutzte Adressraum (Arbeitsspeicher) wird wieder frei gegeben.
  (`Set wdAnw = Nothing`)

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\Basel.doc"

Sub WordDokumentDrucken()
  'Dim wdAnw As Word.Application
  Set wdAnw = CreateObject("Word.Application")
  wdAnw.Visible = False
  wdAnw.DisplayAlerts = 0
  'Dim oDoc As Word.Document
  Set oDoc = wdAnw.Documents.Open(Pfad, _
     ReadOnly:=True, _
        ConfirmConversions:=False, _
           AddToRecentFiles:=False)
  savEnv = wdAnw.Options.PrintBackground
  wdAnw.Options.PrintBackground = False
  oDoc.PrintOut
  wdAnw.Options.PrintBackground = savEnv
  wdAnw.Quit SaveChanges:=False
  Set wdAnw = Nothing
End Sub
```

(Automatisieren bedeutet für den Programmierer: Vorsehen und Vorkehren.)

Automatisieren - Aber «gentleman-like»!

☺ Sollte es erforderlich sein, wie im oben aufgeführten Beispiel Word-Einstellungen zu verändern, dann vergessen Sie bitte nicht, diese alsbald zurückzustellen. Dies sollte sobald als sinnvoll, spätestens aber vor Ende de Routine geschehen. Berücksichtigen Sie dabei alle Ausgänge Ihres Programms, also auch Fehlerroutinen, welche auf *Exit Sub* enden. Alles andere wäre für den Benutzer lästig und irritierend. Er könnte zum Esoteriker vertieren und beginnen zu glauben, sein Rechner hätte ein Innenleben.

Von MS Word auf Access zugreifen

▲ Zurück zum Anfang

Einleitung

Im Zusammenhang mit einem Fernzugriff ist das visuelle Öffnen der Access-Anwendung wohl in den wenigsten Fällen gefragt. Es geht vielmehr um die Daten als um die Oberfläche der Access-Anwendung. In diesem Sinne tritt die *CreateObject*- und *GetObject*-Methode in den Hintergrund, obwohl diese Methoden ebenfalls verwendet werden könnten, wie das letzte Beispiel in diesem Kapitel demonstriert.

Bei den nachfolgenden Beispielen werden allerdings andere Techniken, welche Microsoft zur Verfügung stellt, angewandt. Dabei handelt es sich in zwei Beispielen um **DAO** (*Data Access Object*) und in einem weiteren Beispiel um **ADO** (*ActiveX Data Object*). Die letztere Methode ist moderner, ressourcenschonender und weniger proprietär (offener) als DAO. Allerdings unterstützt MS Word ADO leider noch nicht in allen Teilen.

Alle drei Beispiele beziehen sich auf eine MS Access Datenbank, welche Sie als Beispiel auf Ihrer Office 97 CD finden können. Die Datei hat den Namen **Nordwind.mdb**. In der Office2000-Ausgabe ist diese mdb auch vorhanden, allerdings unter einem etwas anderen Namen.

Diese Datenbank besteht aus diversen Tabellen. Für unsere Beispiele habe ich die Tabelle mit dem Namen **Bestellungen** ausgewählt. Diese Tabelle wiederum beinhaltet ein kleines überschaubares Set an Datenfelder und etwa 80 Datensätze

Eine weitere Gemeinsamkeit der Code-Beispiele ist es, dass jeweils eine sogenannte **SQL-Anweisung** zur Anwendung kommt. SQL ist das Akronym für *Structered Query Language*. Es handelt sich dabei glücklicherweise um eine herstellerunabhängige Methode, welche weite Verbreitung findet. Sie werden darüber - z.B. auch im WEB - sehr viel Literatur finden, um die Sache noch zu verfeinern. Sie können mit einer einzigen SQL-Anweisung eigentlich alles notwendige mitgeben, namentlich:

- Den Namen der **Tabelle**, welche Sie ansprechen möchten.
- Die **Feldinhalte** dieser Tabelle, welche Sie zurück haben möchten.
- Einfache oder auch anspruchsvolle **Abfrage-Optionen**.
- Letztlich können Sie auch noch gleich die **Sortierreihenfolge** der Datensätze vorschreiben..

Mit «DAO» auf Access zugreifen

▲ Zurück zum Anfang

Für das folgende Beispiel gilt folgende Aufgabenstellung:

- In einem neuen Word-Dokument sollen Bestellungen aufgelistet werden, welche sich in der Access-Datenbank mit dem Namen **Nordwind.mdb** befinden. Diese Bestellungen sind in der Datenbank in einer Tabelle mit dem Namen **Bestellungen** untergebracht.
- Allerdings sollen nur jene Bestellungen aufgelistet werden, welche im Feld **Bestimmungsland** den Eintrag *Deutschland* aufweisen.
- Die Datensätze sollen **sortiert** aufgelistet werden und zwar in erster Linie nach dem Wert im Feld *Empfänger* und in zweiter Linie nach dem Feld *Lieferdatum*.
- Bei dieser Auflistung beschränken wir uns auf die beiden Felder mit den Namen *Lieferdatum* und *Empfänger*. Der Asterisk (*****) in der SQL-Anweisung bedeutet übrigens, dass sämtliche Felder zurückgegeben werden. Aus der Fülle dieser Feldwerte, werden aber für unser Beispiel nur die genannten Werte in das Word-Dokument geschrieben.

Implementierung des Beispieles

Um das Beispiel einzurichten, gehen Sie bitte wie folgt vor:

1. Beschaffen Sie sich die Access-Datenbank **Nordwind,** welche Sie als Beispiel auf Ihrer Festplatte oder auf Ihrer Office-CD vorfinden.
2. Wechseln Sie mit der Tastenkombination [**Alt + F11]** in die VBA-Umgebung.
3. Dort angekommen, wählen Sie vom Hauptmenü den ***Dialog Extras - Verweise...***.
4. In der erscheinenden Liste aktivieren Sie den Verweis auf die Objektbibliothek mit dem Namen **Microsoft DAO 3.x Object Library**. Sollte die Liste mehrere Versionen von DAO anzeigen, wählen Sie die neuere sprich höhere Version.
5. Kopieren Sie den folgenden VBA-Code in ein neues Modul, passen Sie allenfalls Pfad und Name der *mdb* im Kopf des Codes noch an und testen Sie das Programm.

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\Nordwind.mdb"

Sub DatenVonAccessImportierenMitDAO1()
  Dim ws As DAO.Workspace
  Dim db As DAO.Database
  Dim rs As DAO.Recordset
  Dim nDoc As Word.Document
  Set ws = DBEngine.Workspaces(0)
  Set db = ws.OpenDatabase(Pfad)
  strSQL = "Select * from Bestellungen"
  strSQL = strSQL & " where Bestimmungsland = 'Deutschland'"
  strSQL = strSQL & " Order by Empfänger, Lieferdatum"
  Set rs = db.OpenRecordset(strSQL, dbOpenSnapshot)
  If rs.RecordCount = 0 Then
    MsgBox "Es wurden keine Entsprechungen gefunden."
    Exit Sub
  End If
  Set nDoc = Documents.Add
  Do While Not rs.EOF
    nDoc.Content.InsertAfter rs.Fields("Lieferdatum") & vbTab _
     & rs.Fields("Empfänger") & vbCrLf
    rs.MoveNext
  Loop
  rs.Close
  db.Close
  Set rs = Nothing
  Set ws = Nothing
  Set db = Nothing
End Sub
```

(Zugriff auf eine Access-Datenbank mittels DAO)

Das nachfolgende Beispiel stellt eine Alternative zum bereits besprochenen Beispiel dar. Es gelten die selben Vorbereitungsarbeiten, wie soeben beschrieben wurden. Bewusst habe ich dieses Beispiel auf das Allerwesentlichste abgespeckt. Damit dürfte gewährleistet sein, dass es auch in einer minimal eingerichteten Umgebung klappen sollte. Der Verweis auf die **DAO-Objektbibliothek** muss allerdings bestehen. Wie dies zu bewerkstelligen ist, wurde bereits erwähnt.

```
Private Const Pfad = "C:\Eigene Dateien\Nordwind.mdb"

Sub DatenVonAccessImportierenMitDAO2()
  Dim db As DAO.Database
  Dim rs As DAO.Recordset
  Set db = OpenDatabase(Name:="Nordwind.mdb", ReadOnly:=True)
  Set rs = db.OpenRecordset(Name:="Bestellungen")
  If rs.RecordCount = 0 Then
    MsgBox "Es wurden keine Entsprechungen gefunden."
    Exit Sub
  End If
  Set nDoc = Documents.Add
  Do While Not rs.EOF
    nDoc.Content.InsertAfter rs.Fields("Bestimmungsland") & vbCrLf
    rs.MoveNext
  Loop
  rs.Close
  db.Close
  Set rs = Nothing
  Set db = Nothing
End Sub
```

(Eine Alternative zum vorhergegangen Beispiel. Auch in diesem Code wird DAO eingesetzt.)

Mit «ADO» auf Access zugreifen

▲ Zurück zum Anfang

Einleitung

Wie schon erwähnt, ist auch der Zugriff mittels ADO auf MS Access möglich. Es scheint, dass diese Methode sogar zusätzlichen Komfort für den Programmierer bereithält. Jedenfalls gefällt mir folgendes Beispiel ausserordentlich gut. Mit sehr wenig Code, lässt sich ein Ergebnis erzielen, welches dies eines Beispiels eigentlich weit übertrifft.

Dies ist einerseits dadurch möglich, dass ADO bei geschicktem Einsatz bereits die Daten von beliebig vielen Datensätze (alle jene welche den Abfrageoptionen genügen) in strukturierter Form rüberschickt. Die *ConvertToTable-Methode* in Word trägt dann noch das ihrige bei, dass auf performande Art und Weise ein zweckmässiges und beschauliches Resultat heraus schaut.

Das Ergebnis des Codes ist eine schön formatierte Tabelle, welche alle Datensätze in der Tabelle **Bestellungen** anzeigt. Aufgelistet werden folgende Feldinhalte: *Lieferdatum*, *Empfänger*, *Ort* und *Bestimmungsland*. Sie können diese Auswahl selbstverständlich Ihren Bedürfnissen anpassen. Einmal mehr, benutzt das Beispiel die Access-Datenbank Nordwind als Quelle.

Implementierung des Beispieles
Um das Beispiel einzurichten, gehen Sie bitte wie folgt vor:

1. Beschaffen Sie sich die Access-Datenbank **Nordwind**, welche Sie als Beispiel auf Ihrer Festplatte oder auf Ihrer Office-CD vorfinden.
2. Wechseln Sie mit der Tastenkombination [**Alt + F11]** in die VBA-Umgebung.
3. Dort angekommen, wählen Sie vom Hauptmenü den Dialog ***Extras - Verweise...***.
4. In der erscheinenden Liste aktivieren Sie den Verweis auf die Objektbibliothek mit dem Namen **Microsoft ActiveX Data Objects x.x Object Library**. Sollte die Liste mehrere Versionen von ADO anzeigen, wählen Sie die neuere sprich höhere Version.
5. Kopieren Sie den folgenden VBA-Code in ein neues Modul, passen Sie allenfalls Pfad und Name der mdb im Kopf des Codes noch an und testen Sie das Programm.

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\Nordwind.mdb"

Sub DatenVonAccessImportierenMitADO()
  Dim oWord As Word.Application
  Dim oDoc As Word.Document
  Dim oRange As Word.Range
  Dim oConn As ADODB.Connection
  Dim oRS As ADODB.Recordset
  Set oWord = CreateObject("Word.Application")
  oWord.Visible = True
  Set oDoc = oWord.Documents.Add
  Set oRange = oDoc.Range
  Set oConn = CreateObject("ADODB.Connection")
  oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
     Pfad & ";Persist Security Info=False"
  Set oRS = oConn.Execute( _
    "SELECT Lieferdatum, Empfänger, Ort, Bestimmungsland FROM Bestellungen")
  sTemp = oRS.GetString(adClipString, -1, vbTab)
  sTemp = "Lieferdatum" & vbTab & "Empfänger" & vbTab & _
     "Ort" & vbTab & "Bestimmungsland" & vbCrLf & sTemp 'Tabellenkopf
  oRange.Text = sTemp
  oRange.ConvertToTable Separator:=vbTab, Format:=wdTableFormatColorful2
  Set oWord = Nothing
  Set oDoc = Nothing
  Set oConn = Nothing
  Set oRS = Nothing
End Sub
```

(Zugriff auf eine Access-Datenbank mittels ADO)

Aus Word ein Makro in Access starten

▲ Zurück zum Anfang

Folgender Code startet z.B. aus MS Word ein Makro in Access. Das gezeigte Beispiel geht davon aus, dass es eine Access-Datenbank mit dem Namen **db1.mdb** gibt. In dieser Datei existiert in einem Modul, dessen Name nicht relevant ist, einen Prozess mit dem Namen **MeineAccessRoutine**.

```
Private Const Pfad = "C:\Eigene Dateien\db1.mdb"

Sub ComboBoxKategorieChange()
  Dim accAnw As Object, accDB As Object
  Set accDB = GetObject(Pfad)
  Set accAnw = accDB.Parent
  accAnw.Run "MeineAccessRoutine", "Anybody there?"
  Set accDB = Nothing
  Set accAnw = Nothing
End Sub
```

(Ein Makro in Access wird z.B. aus Word aufgerufen und es wird auch ein Parameter übergeben...)

Der dazugehörige Prozess in Access lautet für unseres Beispiel dann wie folgt:

```
Sub MeineAccessRoutine(Anfrage As String)
  MsgBox Anfrage & vbCrLf & "This is your Access speaking!"
End Sub
```

(...die Antwort von Access erfolgt prompt.)

Anmerkung zum Code

☹ Meines Wissens unterstützt die Run-Methode das Übergeben von Parameters erst ab **Office2000**. Ohne Parameterübergabe können Sie die Run-Methode aber auch in **Office 97** verwenden.

Mit *«DAO»* Daten aus einem Excel-Tabellenblatt ziehen, ohne die Arbeitsmappe zu öffnen

▲ Zurück zum Anfang

Dank *DAO* ist es möglich, Daten auf performande Art und Weise aus Excel heran zu ziehen. Dabei muss die Excel-Arbeitsmappe nicht mit einer trägen Office-Anwendung geöffnet werden.

Die Kennzeichnung, welche Daten übertragen werden sollen, geschieht in dem Sie einen benannten Bereich in der Excel-Tabelle definieren.

Im Folgenden wird Schritt für Schritt erklärt, wie Sie konkret vorgehen sollten.

Implementierung des Beispieles

1. Öffnen Sie die Excel-Mappe mit den Daten, welche im Listenfeld angezeigt werden sollen.
2. Markieren Sie jene Spalte im Tabellenblatt, welche die Daten für das Listenfeld liefern soll.
3. Vom Hauptmenü in Excel wählen Sie ***Einfügen - Name - Definieren...***.
4. Vergeben Sie nun dem Bereich einen aussagekräftigen Namen. Speichern Sie dann die Excel-Mappe. Sie können die Excel-Anwendung jetzt schliessen.
5. Erstellen Sie nun ein neues Word-Dokument.
6. Wechseln sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
7. Wählen Sie dort vom Hauptmenü ***Einfügen - UserForm***.
8. Fügen Sie der UserForm ein Listenfeld mit dem Namen **ListBox1** bei.
9. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Eintrag **Code anzeigen**.
10. Löschen die zwei Codezeilen, welche Word automatisch generiert hat und kopieren Sie den unten abgedruckten Code in das nun leere Code-Fenster.
11. Passen Sie nun im Kopf des Codes die Konstanten nach Ihren Gegebenheiten an. Die Konstante **Pfad** soll Pfad und Name Ihrer Excel-Arbeitsmappe tragen; die Konstante **Bereich** soll den Bereichsnamen tragen, welchen Sie der fraglichen Spalte in Excel vergeben haben. Abschliessend tragen Sie n die Konstante **StartZeile** den Wert **2** ein, falls die Spalte in Excel eine einzeilige Überschrift trägt, die Daten also auf Zeile 2 beginnen.
12. Immer noch im VBA-Editor klicken Sie nun im Hauptmenü auf ***Extras - Verwiese...***
13. Suchen Sie in der Liste nach einem Eintrag **Microsoft DAO** und aktivieren Sie diesen Verweis. Falls Sie mehrere Einträge bezüglich DAO vorfinden, aktivieren Sie die höchste (sprich neuste) Version
14. Speichern Sie nun das Werk und rufen Sie dann die UserForm auf.

VBA-Code

```
Private Const Pfad = "C:\Eigene Dateien\Kantone der Schweiz.xls"
Private Const Bereich = "Hauptort"
Private Const StartZeile = 2

Private Sub UserForm_Initialize()
  Dim db As DAO.Database
  Dim tblObj As DAO.TableDef
  Dim rs As DAO.Recordset
  Set db = OpenDatabase(Pfad, False, True, "Excel 8.0;")
  DoEvents
  Q = Chr(34)
  Set rs = db.OpenRecordset("Select * FROM `" & Bereich & "`")
  Dim x() As String
  i = -1
  While Not rs.EOF And rs.Fields(0).Value <> ""
    If rs.RecordCount >= StartZeile - 1 Then
      i = i + 1
      ReDim Preserve x(i)
      x(i) = rs.Fields(0).Value
    End If
    rs.MoveNext
  Wend
  ListBox1.List = x()
  Set rs = Nothing
  Set tblObj = Nothing
  Set db = Nothing
End Sub

Private Sub ListBox1_Click()
  MsgBox "Der Benutzer hat " & ListBox1.Value & " gewählt."
End Sub
```

(Ein Listenfeld auf einer UserForm z.B. in MS Word wird mit den Einträgen versorgt, welche aus einer Spalte in einem Excel-Tabellenblatt gezogen werden. Die Mimik ist sehr performand, da die Excel-Arbeitsmappe nicht wirklich geöffnet werden muss.)

Von MS Word auf MS Outlook2000 zugreifen

▲ Zurück zum Anfang

Voranbemerkungen

Die folgenden Beispiele werden erläutern, wie Sie von MS Word auf MS Outlook zugreifen können.

> ! Bitte bringen Sie MS Outlook (es gibt davon die Versionen 97/98 und 2000) nicht mit Outlook Express 4 oder 5 durcheinander. Bei MS Outlook handelt es sich um eine Komponente des MS Office-Paketes. Dieses Paket ist kostenpflichtig. Dagegen ist Outlook Express integrierter Bestandteil des Internet Explorers und wird von Microsoft über verschiedene Kanäle gratis bzw. als Zusatz von Windows vertrieben. Outlook Express ist weit weg von jeder Norm. Ein externer Zugriff ist so gut wie unmöglich; die entsprechenden Schnittstellen sind schlicht und ergreifend nicht vorhanden.

Die Beispiele, welche hier erörtert werden, beschränken sich auf den Zugriff auf den Kontakte-Ordner in MS Outlook. Der Zugriff z.B. auf die Aufgaben ist aber sehr ähnlich, bzw. gleich.

Mittels Office-Automation auf den Kontakte-Ordner von MS Outlook zugreifen

Direkter Zugriff mittels Schlüssel

Das nachfolgende Beispiel fragt Sie nach dem Schlüssel (Namen) eines Kontaktes und zeigt dann die Geschäftstelefonnummer dieses Datensatzes in einer Meldungs-Box an.

Implementierung des Beispieles

Um das Beispiel zu testen, gehen Sie bitte wie folgt vor:

1. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
2. Dort angekommen, wählen Sie vom Hauptmenü den ***Dialog Extras - Verweise...***.
3. In der erscheinenden Liste aktivieren Sie den Verweis auf die Objektbibliothek mit dem Namen ***Microsoft Outlook 9.0 Object Library***.
4. Kopieren Sie den folgenden VBA-Code in ein neues Modul. Passen Sie die Konstante mit dem Namen **Schluessel** im Kopf des Codes an und testen Sie das Programm.

VBA-Code

```
Private Const Schluessel = "Berta Maus" 'Ohne Komma!!!
Private Const Titel = "Telefonnummer von "

Sub ZugriffAufKontakeOrdnerPerSchluessel1()
  Dim olOrdner As Outlook.MAPIFolder
  Dim olKontakt As Outlook.ContactItem
  Dim olAnw As Outlook.Application
  'Unsichtbare Instanz von MS Outlook starten
  Set olAnw = CreateObject("Outlook.Application")
  'Kontakte-Ordner setzen
  Set olOrdner = olAnw.Session.GetDefaultFolder(olFolderContacts)
  On Error Resume Next
  Set olKontakt = olOrdner.Items(Schluessel)
  If Err.Number > 0 Then 'Allgemeiner Fehler
    MsgBox Err.Description
    Exit Sub
  End If
  On Error GoTo 0
  If olKontakt Is Nothing Then
    MsgBox "Der Schlüssel wurde nicht gefunden.", vbExclamation, _
        Titel & Schluessel
  Else
    MsgBox olKontakt.BusinessTelephoneNumber, vbInformation, _
       Titel & Schluessel
  End If
  Set olKontakt = Nothing
  Set olOrdner = Nothing
  Set olAnw = Nothing
End Sub
```

(Zugriff auf den Kontakte-Ordner von MS Outlook mittels Schlüssel)

Sie können diese und andere Übungen auch durchführen, ohne vorgängig einen Verweis auf die Objektbibliothek von MS Outlook zu setzen. Der Deklarationsteil des Codes muss dann aber anders aussehen. Bedenken Sie, dass Sie sich in diesem Fall folgende Eigenarten bzw. Nachteile einhandeln:

- Statt einer spezifischen Objekt-Deklaration, werden alle MS Outlook-Objekte als **Object** deklariert.
- Eigenschaften und Methoden von allgemein deklarierten MS Outlook-Objekten werden beim Entwickeln nicht mehr als **Quick Info** angezeigt. Sie müssen diese Information zuerst in der VBA-Hilfe von MS Outlook nachschlagen.
- Sie können keine MS Outlook-**Konstanten** mehr über Ihren Namen verwenden. Stattdessen müssen Sie den absoluten Wert der Konstante in der Hilfe von MS Outlook nachschlagen. Aus *olFolderContacts* z.B. wird dadurch die nichts aussagende Zahl 10.

Hier also der entsprechende VBA-Code, welche keinen Verweis auf die MS Outlook-Objektbibliothek voraussetzt:

```
Private Const Schluessel = "Berta Maus" 'Ohne Komma!!!
Private Const Titel = "Telefonnummer von "

Sub ZugriffPerSchluessel2()
  Dim olOrdner As Object
  Dim olKontakt As Object
  Dim olAnw As Object
  'Unsichtbare Instanz von MS Outlook starten
  Set olAnw = CreateObject("Outlook.Application")
  'Kontakte-Ordner setzen
  Set olOrdner = olAnw.Session.GetDefaultFolder(10)
  On Error Resume Next
  Set olKontakt = olOrdner.Items(Schluessel)
  If Err.Number > 0 Then 'Allgemeiner Fehler
    MsgBox Err.Description
    Exit Sub
  End If
  On Error GoTo 0
  If olKontakt Is Nothing Then
    MsgBox "Der Schlüssel wurde nicht gefunden.", vbExclamation, _
        Titel & Schluessel
    Exit Sub
  End If
  MsgBox olKontakt.BusinessTelephoneNumber, vbInformation, _
     Titel & Schluessel
  Set olKontakt = Nothing
  Set olOrdner = Nothing
  Set olAnw = Nothing
End Sub
```

(Zur Not geht es auch ohne Verweis, auf die MS Outlook-Objektbibliothek.)

Zugriff Scan-Ergebnistabelle

Im nächsten Beispiel wird nach allen Kontakten gefragt, deren Firma den Wert **Novartis** trägt. Diese Kontakte werden in einer Word-Tabelle dargestellt.

Implementierung des Beispieles

Um das Beispiel zu testen, gehen Sie bitte wie folgt vor:

1. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
2. Dort angekommen, wählen Sie vom Hauptmenü den Dialog ***Extras - Verweise...***.
3. In der erscheinenden Liste aktivieren Sie den Verweis auf die Objektbibliothek mit dem Namen **Microsoft Outlook 9.0 Object Library**.
4. Kopieren Sie den folgenden VBA-Code in ein neues Modul. Passen Sie die Konstante mit dem Namen **Schluessel** im Kopf des Codes an und testen Sie das Programm.

VBA-Code

```
Sub ZugriffPerScan()
  Dim olAnw As Outlook.Application
  Dim olOrdner As Outlook.MAPIFolder
  Dim olKontakte As Outlook.Items
  Dim olKontakt As Outlook.ContactItem
  'Unsichtbare Instanz von MS Outlook starten
  Set olAnw = CreateObject("Outlook.Application")
  'Kontakte-Ordner setzen
  Set olOrdner = olAnw.Session.GetDefaultFolder(olFolderContacts)
  Set olKontakte = olOrdner.Items
  'Scan ansetzen
  Set olKontakte = olKontakte.Restrict("[CompanyName] = ""Novartis""")
  If olKontakte.Count = 0 Then
    MsgBox "Es wurde keine Entsprechung gefunden."
    Exit Sub
  End If
  'Kopfzeile setzen
  strText = "Nachname" & vbTab & "Vorname" & vbTab & "Telefonnummer" & vbCrLf
  'Scan-Result-Liste sequentiell auslesen
  For Each olKontakt In olKontakte
    strText = strText & _
       olKontakt.LastName & vbTab & _
          olKontakt.FirstName & vbTab _
             & olKontakt.BusinessTelephoneNumber & vbCrLf
  Next
  Dim oDoc As Word.Document
  Set oDoc = Documents.Add
  oDoc.Content = strText
  oDoc.Content.ConvertToTable Separator:=vbTab, _
     Format:=wdTableFormatClassic2, AutoFit:=True
  oDoc.Tables(1).Sort ExcludeHeader:=True
  MsgBox olKontakte.Count & " Addressen aufgelistet."
  Set olKontakt = Nothing
  Set olKontakte = Nothing
  Set olOrdner = Nothing
  Set olAnw = Nothing
End Sub
```

(Zugriff auf den Kontakte-Ordner von MS Outlook mittels "Scan".)

Zugriff auf enumerierte Felder

MS Outlook kennt auch Felder, welche keinen einzelnen festen Wert enthalten, sondern eine Aufzählung von vordefinierten Stichwörter. Das wohl meist gebrauchte Feld dieser Klasse ist das Feld mit dem Namen **Kategorie**.

Die *Restrict*-Methode ist im allgemeinen sehr praktisch und einfach in der Anwendung, versagt aber ihren Dienst bei enumerierten Feldern. Hier bleibt Ihnen wohl nur, diese Sache selbst auszuprogrammieren. Dabei können Sie sich am folgenden Beispiel orientieren.

Aufgabenstellung
Das Beispiel geht davon aus, dass Ihr Kontakt-Ordner einige Kontakte enthält, bei denen im Feld Kategorien u.a. der Feldwert *Hausfreund* enthalten ist. Dieser Wert soll jetzt bei allen Datensätzen in *Lover* geändert werden. Zusätzliche Werte, welche in den selben Datensätzen ebenfalls vorhanden sein können, dürfen dabei nicht tangiert werden.

Hier ein Beispiel: Enthält ein Datensatz (Kontakt) im Feld Kategorien folgende Aufzählung: *Privat, Hausfreund, Kollege* dann soll das nach *Privat, Lover, Kollege* geändert werden. Dies soll gleich bzw. analog für alle Datensätze geschehen, welche im besagten Feld nur oder auch den Begriff Hausfreund enthalten.

Implementierung
Es gelten die selben Vorbereitungsarbeiten, wie bei den bereits besprochenen Beispielen.

VBA-Code

```
Sub ZugriffAufFeldKategorien()
  Suche = "Hausfreund"
  ErsetzeMit = "Lover"
  Dim Ordner As Outlook.MAPIFolder
  Dim Kontakte As Outlook.Items
  Dim Kontakt As Outlook.ContactItem
  'Kontakte-Ordner setzen
  Set Ordner = Session.GetDefaultFolder(olFolderContacts)
  Set Kontakte = Ordner.Items
  If Kontakte.Count = 0 Then
    MsgBox "Es wurde keine Entsprechung gefunden."
    Exit Sub
  End If
  AnzMutationen = 0
  For Each Kontakt In Kontakte
    i = i + 1
    If InStr(Kontakt.Categories, Suche) Then
      Dim tmp() As String
      tmp() = Split(Kontakt.Categories, ";")
      For i = 0 To UBound(tmp())
        If tmp(i) = Suche Then
          tmp(i) = "Lover"
          AnzMutationen = AnzMutationen + 1
        End If
      Next i
      Kontakt.Categories = Join(tmp(), ";")
      Kontakt.Save
    End If
  Next
  MsgBox AnzMutationen & " Mutationen getätigt."
  Set Kontakt = Nothing
  Set Kontakte = Nothing
  Set Ordner = Nothing
End Sub
```

(Die *Restrict*-Methode ist auf enumerierte Felder, wie zum Beispiel das Feld Kategorie nicht applizierbar. Somit muss Ihr Code alle Datensätze auslesen und vergleichen.)

Anmerkung zum Code

! Bitte beachten Sie, dass die **Split-** und **Join** Anweisung erst ab **Office2000** zu Ihrer Verfügung steht.

MS Word Seriendruck aus dem Internet Explorer automatisiert ausführen

▲ Zurück zum Anfang

Einleitung

Zum Abschluss dieses Skriptes noch ein kleiner Höhepunkt. Die Aufgabenstellung dabei ist es, aus dem Internet Explorer - also einer nicht MS-Office Anwendung - MS Word unsichtbar im Hintergrund zu starten und diese Anwendung dazu veranlassen über die Seriendruckfunktion auf einen vorgegebenen Drucker, Etiketten auszudrucken. Dies soll alles vollautomatisch stattfinden. Der Benutzer, welcher dies auslöst, spürt gar nichts davon, dass MS Word und Office-Automation im Spiel ist.

Ob die Quelldaten für die Etiketten in Access, Excel oder wo auch immer geführt werden, ist nur am Rande Teil des Automatisationsprozesses. Dies ist letztlich im Seriendruckhauptdokument festgesetzt, welches vorgängig in MS Word eingerichtet wird.

Erreicht wird das Ziel dank einem kleinen VBScript, welches Sie an beliebiger Stelle in ein HTML-Dokument einbauen können. Der Benutzer sieht dann eine graue Schaltfläche, Dank welcher er die Sache in Gang bringen kann.

Implementierung des Beispieles

Um das Beispiel in Gang zu setzen, gehen Sie bitte wie folgt vor:

1. Starten Sie Word und wählen Sie vom Hauptmenü den Dialog ***Extras - Seriendruck - Erstellen - Adressetiketten... - Aktives Fenster***.
2. Über den Schaltknopf **Daten importieren** wählen Sie nun die Datenquelle aus.
3. Klicken Sie nun auf den Schaltknopf **Hauptdokument einrichten**.
4. Wählen Sie im Nachfolgenden Dialog das passende Etikett aus.
5. Im nun nachfolgenden Dialog setzen Sie die gewünschten Seriendruckfelder ein und verlassen den Dialog über die Schaltfläche Schliessen.
6. Wechseln Sie nun in diesem Serienbriefhauptdokument mittels der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
7. Fügen sie dem Projekt ein neues Modul zu und kopieren Sie in dieses Modul den VBA-Code, welcher unten angezeigt ist.
8. Passen Sie im Kopf des Codes die Konstante mit dem Namen **Drucker** Ihren Gegebenheiten an.
9. Speichern Sie nun das Dokument und verlassen Sie Word.
10. Mit dem kleinen Hilfsprogramm *Notepad.exe* erstellen Sie eine Datei und speichern diese Datei unter einem beliebigen Namen, allerdings mit der Dateierweiterung **.htm**.
11. Kopieren Sie das unten abgebildete VBScript in diese HTML-Datei. Vergessen Sie nicht Pfad und Name des Serienbriefhauptdokumentes anzupassen.
12. Führen sie dann einen Doppelklick auf diese Datei aus und betätigen Sie den Schaltknopf Etiketten erstellen.
13. Sie sollten nun feststellen können, wie die Festplatte arbeitet und nach einigen Sekunden, sollten die Etiketten auf Ihrem Drucker gedruckt werden.

VBA-Code

```
Private Const Drucker = "Acrobat PDFWriter"

Sub AutoOpen()
  savEnv = ActivePrinter
  ActivePrinter = Drucker
  With ActiveDocument.MailMerge
    .Destination = wdSendToNewDocument
    .Execute
  End With
  ActiveDocument.PrintOut Background:=False
  ActivePrinter = savEnv
  ActiveDocument.Close SaveChanges:=False
End Sub
```

(Kopieren Sie diesen Code im VBA-Editor in ein neues Modul des Seriendruckhauptdokument und passen Sie den Namen des Druckers im Kopf des Codes an)

VBScript für die HTML-Datei

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE=VBScript>

Dim objWord

Sub Btn1_onclick()
  call OpenDoc("C:\Eigene Dateien/AASerEtt.doc")
End Sub

Sub OpenDoc(strLocation)
  Set objWord = CreateObject("Word.Application")
  objWord.Documents.Open strLocation
  objWord.Visible = False
  objWord.Quit False
End Sub

</SCRIPT>

<TITLE>Ettiketten Drucken</Title>
</HEAD>

<BODY>

<INPUT TYPE=BUTTON NAME=Btn1 VALUE="Etiketten drucken">

</BODY>
</HTML>
```

(Kopieren Sie diesen Code zuerst zum Testen in eine neue HTML-Datei und später in Ihre HTML-Datei. Sie müssen den Namen der Word-Datei noch anpassen.

Anmerkung

- Abhängig davon, welche Sicherheitsstufe Sie im Internet Explorer eingestellt haben, erfolgt vor Ausführung des VBScriptes einen (oder gleich mehrere) Sicherheitshinweis(e) an den Benutzer.

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Office Automation

Microsoft

► Microsoft Office 97 Automation Help File

► OFF2000: Microsoft Office2000 Automation Help File Available

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/OA.htm

René Probst, Oktober 1999

Änderungen

- 5. Mai 2000 - Excel Tabellenblatt in Datenfeld laden
- 5. Mai 2000 - Word ex Excel starten
- 5. Mai 2000 - Verweise auf Objektbibliotheken
- 17. Mai 2000 - Office 97 und 2000 Hilfe-Dateien
- 11. Oktober 2000 - Zusätzliches für Excel und Word
- 11. Oktober 2000 - Beispiel für Access
- 11. Oktober 2000 - Beispiel für MS Outlook
- 11. Oktober 2000 - Beispiel für VBSkript ex Internet Explorer
- 17. Dezember 2000 - Excel-Daten mittels DAO anziehen
- 17. Januar 2001 - Excel mit Vorlage (XLT) starten

Wie lese oder setze ich mit VBA den Wert einer Dokumenteneigenschaft?

◄ Zurück zum Hauptindex

Problem

> Mittels VBA sollte ich Dokumenteneigenschaften, wie sie im Dialog *Datei - Eigenschaften* erscheinen, lesen bzw. setzen. Wie lautet der entsprechende Code?

Diese Seite

- Mittels VBA auf Dokumenteigenschaften zugreifen
- Tabelle der word-integrierten Dokumenteneigenschaften
- Benutzerdefinierte Dokumenteneigenschaften
- Dokumenteneigenschaften offline lesen
- Auf die Vorschaugrafik eines Office-Dokumentes zugreifen
- Feststellen, ob ein Dokument Makros enthält
- Feststellen in welcher Version ein Office-Dokument erstellt wurde
- Inventar der Office-Dokumente erstellen
- Die Arbeit von David Foster
- Links zu diesem Thema

Mittels VBA auf Dokumenteigenschaften zugreifen

▲ Zurück zum Anfang

Um eine Dokumenteneigenschaft anzusprechen benötigen Sie folgenden Code:

```
Autor = ActiveDocument.BuiltInDocumentProperties(wdPropertyAuthor).Value
```

(Die word-integrierte Dokumenteneigenschaft "Autor" wird ausgelesen.)

```
ActiveDocument.BuiltInDocumentProperties(wdPropertyAuthor).Value = "Buchhalter
Klötzli"
```

(Die word-integrierte Dokumenteneigenschaft "Autor" wird gesetzt.)

Anmerkungen

- Um diese Dokumenteneigenschaften anzusprechen, brauchen Sie entweder den Namen der entsprechenden englischsprachigen Word-Variablen, oder die entsprechende **ID**. Bei der ID handelt es sich um eine integere Zahl zwischen 1 und 30. Statt *wdPropertyAuthor* könnten Sie auch die ID 3 benutzen. Die Syntax sieht dann wie folgt aus:
  `Autor = ActiveDocument.BuiltInDocumentProperties(3).Value`
- Die Beziehung zwischen den deutschen Eigenschaftsnamen wie sie im Dialog *Datei - Eigenschaften* erscheinen, den englischen wd-Variablen und der ID entnehmen Sie bitte der nachfolgenden Tabelle.

Tabelle der word-integrierten Dokumenteneigenschaften

▲ Zurück zum Anfang

ID	Word-Konstante	Deutsche Bezeichnung
1	wdPropertyTitle	Titel
2	wdPropertySubject	Thema
3	wdPropertyAuthor	Autor
4	wdPropertyKeywords	Stichwörter
5	wdPropertyComments	Kommentar
6	wdPropertyTemplate	Vorlage
7	wdPropertyLastAuthor	Zuletzt gespeichert von
8	wdPropertyRevision	Version
9	wdPropertyAppName	nicht verfügbar
10	wdPropertyTimeLastPrinted	Gedruckt am
11	wdPropertyTimeCreated	Erstellt am
12	wdPropertyTimeLastSaved	Geändert am
13	wdPropertyVBATotalEdit	Auf Druck der Gewerkschaften in der deutschen Office Version nicht implementiert
14	wdPropertyPages	Seiten
15	wdPropertyWords	Wörter
16	wdPropertyCharacters	Zeichen (ohne Leerzeichen)
17	wdPropertySecurity	Dokumentenschutz
18	wdPropertyCategory	Kategorie
19	wdPropertyFormat	nicht verfügbar
20	wdPropertyManager	Manager
21	wdPropertyCompany	Firma
22	wdPropertyBytes	Bytes
23	wdPropertyLines	Zeilen
24	wdPropertyParas	Absätze
25	wdPropertySlides	nicht verfügbar (PowerPoint)
26	wdPropertyNotes	nicht verfügbar (PowerPoint)
27	wdPropertyHiddenSlides	nicht verfügbar (PowerPoint)
28	wdPropertyMMClips	nicht verfügbar (PowerPoint)
29	wdPropertyHyperlinkBase	Hyperlinkbasis
30	wdPropertyCharsWSpaces	Buchstaben (mit Leerzeichen)

Benutzerdefinierte Dokumenteneigenschaften

▲ Zurück zum Anfang

Unabhängig zu den word-integrierten Dokumenteneigenschaften, können Sie auf der Ebene eines Dokumentes oder einer Dokumentenvorlage auch eigene Eigenschaften definieren, lesen und setzen. Dies ist sowohl über den Word-Dialog ***Datei - Eigenschaften - Anpassen*** wie auch VBA-technisch möglich.

Benutzerdefinierte Dokumenteneigenschaft mittels Word-Dialog administrieren

Wie schon erwähnt, ist dies über den Dialog ***Datei - Eigenschaften - Anpassen*** möglich. Die Benutzerführung ist eigentlich selbsterklärend.

Benutzerdefinierte Dokumenteneigenschaft mittels VBA erstellen, setzen und lesen

```
ActiveDocument.CustomDocumentProperties.Add LinkToContent:=False, _
 Name:="Abteilung", Type:=msoPropertyTypeString, Value:="Software Entwicklungen"
```

(Benutzerdefinierte Dokumenteneigenschaft erstellen und gleichzeitig einen Wert zuweisen.)

```
Abteilung = ActiveDocument.CustomDocumentProperties("Abteilung").Value
```

(Benutzerdefinierte Dokumenteneigenschaft einlesen.)

```
ActiveDocument.CustomDocumentProperties("Abteilung").Value = "Logisches Netzwerk"
```

(Benutzerdefinierte Dokumenteneigenschaft zuweisen.)

Dokumenteneigenschaften *offline* lesen

▲ Zurück zum Anfang

Immer wieder taucht der Wunsch bzw. die Frage auf, ob es möglich ist, Dokumenteneigenschaften von Word- und anderen MS Office-Dokumenten einzulesen, ohne auf eine träge Office-Anwendung angewiesen zu sein. Inzwischen gibt es gleich zwei Möglichkeiten in diesem Bereich etwas erfolgsversprechendes auf die Beine zu stellen.

Mittels ActiveX-DLL von Microsoft

Microsoft stellt dafür die Programmbibliothek **dsofile.dll** zur Verfügung, welche sich einfach in das VBA-Umfeld eingliedern lässt und es ermöglicht, Dateieigenschaften von MS Office-Dokumenten auf effiziente Art auszulesen.

Wo sie die genannte DLL gratis herunter laden können, erfahren Sie am Schluss dieses Skriptes. Nach dem Entpacken der Datei **dsofile.exe**, finden Sie neben einer *Readme*-Datei und einigen Beispieldateien die erwähnte DLL. Sie brauchen eigentlich einzig diese DLL für ein Gelingen. Zur Implementierung der Programmbibliothek gehen Sie bitte wie folgt vor:

1. Kopieren Sie die Pragrammbibliothek in das Windows-Systemverzeichnis, also z.B. in das Verzeichnis mit dem Namen **C:\Windows\System**.
2. Als nächstes sollte Sie die DLL technisch registrieren. Wählen Sie dazu den Windows-Dialog ***Start - Ausführen***. In der Kommandozeile geben Sie folgendes ein: `regsvr32 dsofile.dll`. Dieser Vorgang liefert eine Endmeldung (hoffentlich eine positive).
3. Starten Sie nun eine Office-Anwendung, also z.B. MS Word.
4. **Öffnen** Sie nun die Standardvorlage **Normal.dot** und wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
5. Wählen Sie dort vom Hauptmenü den Dialog ***Extras - Verweise...***.
6. Betätigen Sie die Schaltfläche ***Durchsuchen...*** und wechseln Sie in das Windows-System-Verzeichnis.
7. Suchen Sie nun nach der Datei **dsofile.dll** und betätigen Sie die Schaltfläche ***Öffnen***.
8. Damit ist ein Verweis auf **DS: OLE Document Properties 1.1 Object Library** erstellt. und die nötigen Objekte, Eigenschaften und Methoden stehen Ihnen nun in VBA zur Verfügung.
9. Speichern Sie nun die "Normal.dot" und verlassen Sie die Anwendung. Word sollte sich diese Einstellung für die Zukunft merken.

Nachfolgend nun die Beispiele, wie Sie die DLL ansprechen bzw. nutzen können. Die Sache ist recht einfach. Lassen Sie sich nicht durch den sich immer wiederholenden Code bluffen. Ich habe mich hier entschieden, für jede mögliche Aktion eine Prozedur zu schreiben, welche in sich eigenständig und lauffähig ist.

```
Sub IntegrierteEigenschaftLesen()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim iProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set iProp = oFilePropReader.GetDocumentProperties(Datei)
  MsgBox iProp.Title
  Set oFilePropReader = Nothing
  Set iProp = Nothing
End Sub
```

(Die integrierte Eigenschaft "Titel", englisch Title, wird ausgelesen.)

```
Sub IntegrierteEigenschaftSetzen()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim iProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set iProp = oFilePropReader.GetDocumentProperties(Datei)
  iProp.Title = "Es war einmal... vor langer Zeit..."
  Set oFilePropReader = Nothing
  Set iProp = Nothing
End Sub
```

(Die integrierte Eigenschaft "Titel", wird gesetzt.)

```
Sub BenutzerdefinierteEigenschaftHinzufügen()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim cProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set cProp = oFilePropReader.GetDocumentProperties(Datei)
  cProp.CustomProperties.Add "Vertretung", "Finnland"
  Set oFilePropReader = Nothing
  Set cProp = Nothing
End Sub
```

(Die benutzerdefinierte Eigenschaft "Vertretung", wird zum Dokument hinzu gefügt. Gleichzeitig wird der Wert auf "Finnland" gesetzt.)

```
Sub BenutzerdefinierteEigenschaftLesen()
  Datei = "C:\Eigene Dateien\Test.doc"  'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim cProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set cProp = oFilePropReader.GetDocumentProperties(Datei)
  MsgBox cProp.CustomProperties.Item("Vertretung")
  Set oFilePropReader = Nothing
  Set cProp = Nothing
End Sub
```

(Die benutzerdefinierte Eigenschaft "Vertretung" wird ausgelesen.)

```
Sub BenutzerdefinierteEigenschaftNeuSetzen()
  Datei = "C:\Eigene Dateien\Test.doc"   'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim cProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set cProp = oFilePropReader.GetDocumentProperties(Datei)
  cProp.CustomProperties.Item("Vertretung") = "Frankreich"
  Set oFilePropReader = Nothing
  Set cProp = Nothing
End Sub
```

(Die benutzerdefinierte Eigenschaft "Vertretung" wird verändert.)

```
Sub BenutzerdefinierteEigenschaftLöschen()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim cProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set cProp = oFilePropReader.GetDocumentProperties(Datei)
  cProp.CustomProperties.Item("Vertretung").Remove
  Set oFilePropReader = Nothing
  Set cProp = Nothing
End Sub
```

(Die benutzerdefinierte Eigenschaft "Vertretung" wird vom Dokument entfernt.)

Benutzerdefinierte Eigenschaften können auch sequentiell ausgelesen werden, ohne dass zum voraus bekannt ist, welche dieser Eigenschaften im Dokument definiert sind.

Dieses Code-Beispiel verdeutlicht auch, dass MS Office fünf verschiedene Typen von Dokumenteneigenschaften kennt, namentlich:

- Zeichenfolge (String)
- Ganzzahl (Integer)
- Gleitkommazahl (Long)
- Wahr/Falsch (Boolean)
- Datum (Date)

Leider habe ich keine Möglichkeit gefunden, auch die integrierten Eigenschaften sequentiell auszulesen. Das ist aber halb so schlimm, weil diese Eigenschaften (im Gegensatz zu den benutzerdefinierten) eigentlich klar deklariert sind und in diesem Skript auch schon namentlich aufgeführt wurden (Siehe Tabelle weiter oben). Sollten Sie trotzdem nicht darauf verzichten können, dann schauen Sie sich bitte die Lösung von David Foster - etwas weiter unten im Skript - an. Mit der Arbeit von David ist es möglich, auch integrierte Dokumenteneigenschaften in einer Schleife auszulesen.

```
Sub BenutzerdefinierteEigenschaftenSequenziellLesen()
  Datei = "C:\Eigene Dateien\Test.doc"  'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim cProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set cProp = oFilePropReader.GetDocumentProperties(Datei)
  Dim x As DSOleFile.CustomProperty
  For Each x In cProp.CustomProperties
    MsgBox x.Name & vbCrLf & x.Value & vbCrLf & EditType(x.Type)
  Next
End Sub

Private Function EditType(tType)
  Select Case tType
    Case 1: EditType = "Zeichenfolge"
    Case 2: EditType = "Ganzzahl"
    Case 3: EditType = "Gleitkommazahl"
    Case 4: EditType = "Wahr/Falsch"
    Case 5: EditType = "Datum"
  End Select
End Function
```

(In diesem Beispiel werden alle benutzerdefinierten Eigenschaften - falls vorhanden - ausgelesen. Die Namen dieser Eigenschaften müssen dabei nicht explizit bekannt sein.)

Auf die Vorschaugrafik eines Office-Dokumentes zugreifen

▲ Zurück zum Anfang

Dank der DLL von Microsoft ist es nun auch möglich, eine im Office-Dokument enthaltene **Vorschaugrafik** auszulesen und anzuzeigen. Es handelt sich dabei um jene Grafik, welche für das Dokument erstellt wird, wenn Sie - auf Word bezogen - Den Dialog ***Datei - Eigenschaften*** aufrufen und dort ein Häkchen bei **Vorschaugrafik speichern** setzen.

Um die Grafik anzuzeigen benötigen wir ein geeignetes Gefäss. Im folgenden Code-Beispiel, habe ich eine ActiveX Komponente gewählt, welche ich direkt im Dokument dynamisch erstelle. Es handelt sich dabei um einen **Bild**-Rahmen, welchen Sie über die Symbolleiste **Steuerelement-Toolbox** auch manuell dem Dokument zufügen können. Selbstverständlich können Sie als Gefäss auch ein Anzeige-Steuerelement auf einer **UserForm** (**Image-Control**) verwenden. Setzen Sie die Eigenschaft **AutoSize** auf **True** oder zeichnen Sie den Bildrahmen genügend gross, damit das Ergebnis auch gut sichtbar ist. Müssig zu erwähnen, dass das Dokument über eine Vorschaugrafik verfügen sollte.

```
Sub VorschauGrafikAnzeigen()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oDoc As Document
  Dim Rahmen As Object
  Dim Bild As StdPicture
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim iProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set iProp = oFilePropReader.GetDocumentProperties(Datei)
  On Error Resume Next
  Set Bild = iProp.Thumbnail
  On Error GoTo 0
  If Not Bild Is Nothing Then
    Set oDoc = Documents.Add
    Set Rahmen = oDoc.InlineShapes.AddOLEControl(ClassType:="Forms.Image.1")
    With Rahmen.OLEFormat.Object
      .AutoSize = True
      .Picture = Bild
    End With
  Else
    MsgBox "Die Datei " & Datei & " enthält keine Vorschaugrafik."
  End If
  Set oFilePropReader = Nothing
  Set iProp = Nothing
End Sub
```

(In diesem Beispiel wird die Vorschaugrafik eines Office-Dokumentes ausgelesen, ohne die Datei zu öffnen. Das Resultat wird in einem ActiveX-Bildrahmen angezeigt.)

Feststellen, ob ein Dokument Makros enthält

▲ Zurück zum Anfang

Geht es darum herauszufinden, ob ein Office-Dokument **Makros** enthält, können Sie folgenden Code verwenden. Microsoft sagt aus, dass die Methode ausschliesslich auf Word- und Excel-Dateien anwendbar ist.

```
Sub FeststellenObDasDokumentMakrosEnthält()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim iProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set iProp = oFilePropReader.GetDocumentProperties(Datei)
  On Error Resume Next
  x = iProp.HasMacros
  If Err.Number > 0 Then
    MsgBox Datei & " ist kein Word- oder Excel-Dokument"
  Else
    If x = 0 Then
      MsgBox "Keine Makros."
    Else
      MsgBox "Makros vorhanden."
    End If
  End If
  On Error GoTo 0
  Set oFilePropReader = Nothing
  Set iProp = Nothing
End Sub
```

(In diesem Beispiel wird eruiert, ob ein Word- oder Excel-Dokument Makros enthält, ohne die Datei zu öffnen.

Folgende Routine stellt fest, ob ein fragliches Office-Dokument schreibgeschützt ist. Dabei ist der Schreibschutz gemeinte, welcher vom Dateisystem des Betriebssystems vergeben wird. Ich habe leider keine Möglichkeit entdeckt, zu eruieren, ob einem Office-Dokument auf Stufe der Office-Anwendung ein Schreibschutz (sprich Passwort) verpasst wurde. Ich glaube auch nicht, dass dies möglich gemacht wird. Man möchte ja potentiellen Cracker, die Arbeit nicht erleichtern.

```
Sub FeststellenObDasDokumentSchreibgeschütztIst()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim iProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set iProp = oFilePropReader.GetDocumentProperties(Datei)
  If iProp.IsReadOnly = True Then
    MsgBox "Das Dokument ist schreibgeschützt."
  Else
    MsgBox "Das Dokument hat keinen Dateisystemschreibschutz."
  End If
  Set oFilePropReader = Nothing
  Set iProp = Nothing
End Sub
```

(In diesem Beispiel wird eruiert, ob das Office-Dokument auf Ebene des Dateisystems schreibgeschützt ist).

Feststellen in welcher Version ein Office-Dokument erstellt wurde

▲ Zurück zum Anfang

Endlich gibt es auch eine halbwegs vernünftige Möglichkeit festzustellen, unter welcher **Office-Version** ein Dokument entstanden ist. VBA bietet dazu sonderbarerweise keine integrierte Funktion. Die entsprechenden Eigenschaften, welche im Office-Objekt-Model zu finden sind, sagen nur aus, welche Office-Programm-Version auf dem Rechner installiert ist. Diese Information ist aber bei vielen Aufgaben nicht von Interesse. Richtig ist, dass sich schon andere Leute in dieser Angelegenheit verdient gemacht haben. Auch diese Lösungen basieren auf den Einsatz einer eigenen Programmbibliothek.

```
Sub FeststellenUnterWelcherWordVersionDasDakumentErstelltWurde()
  Datei = "C:\Eigene Dateien\Test.doc" 'Bitte gültiges Office-Dokument angeben
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim iProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  Set iProp = oFilePropReader.GetDocumentProperties(Datei)
  MsgBox iProp.AppName
  Set oFilePropReader = Nothing
  Set iProp = Nothing
End Sub
```

(Feststellen in welcher Version ein Office-Dokument vorliegt. Diese Funktion bzw. Eigenschaft vermissen wir schmerzlich in VBA)

Inventar der Office-Dokumente erstellen

▲ Zurück zum Anfang

Das folgende Beispiel erstellt eine Tabelle mit allen Word-Dokumenten im Standard-Verzeichnis. Neben dem Dateinamen werden die integrierten Dokumenteneigenschaften **Titel**, **Autor** und **Zuletzt gespeichert** aufgelistet.

Kopieren Sie den nachfolgenden Code im VBA-Editor in ein neues Modul, rufen Sie das Makro mit dem Namen **IntegrierteEigenschaftenEinesVerzeichnisesAuflisten1** auf und bald werden Sie einen guten Überblick über Ihre Word-Dokumente erhalten.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Sub IntegrierteEigenschaftenEinesVerzeichnisesAuflisten1()
  Pfad = Options.DefaultFilePath(wdDocumentsPath)
  With Application.FileSearch
    .NewSearch
    .FileName = "*.doc*"
    .LookIn = Pfad
    .SearchSubFolders = False
    .Execute SortBy:=msoSortByFileName
    AnzDateien = .FoundFiles.Count
    If AnzDateien = 0 Then
      MsgBox Pfad
      End
    End If
    j = 0
    Dim rDoc As Document
    Set rDoc = Documents.Add
    Application.ScreenUpdating = False
    ReDim Datei(AnzDateien, 2)
    For i = 1 To AnzDateien
      tmp = Dir(.FoundFiles(i))
      If tmp <> "" Then
        tmp = Left(tmp, Len(tmp) - 4)
        j = j + 1
        Datei(j, 1) = .FoundFiles(i)
        Datei(j, 2) = tmp
      End If
    Next i
  End With
  AnzDateien = j
  D = Chr(1)
  Dim oFilePropReader As DSOleFile.PropertyReader
  Dim iProp As DSOleFile.DocumentProperties
  Set oFilePropReader = New DSOleFile.PropertyReader
  tstr = "Dateiname" & D & "Titel" & D & "Autor" & D & "Zuletzt gespeichert" & vbCrLf
  Selection.TypeText Text:=tstr
  For i = 1 To AnzDateien
    tstr = Datei(i, 2) & D
    On Error Resume Next
    Set iProp = oFilePropReader.GetDocumentProperties(Datei(i, 1))
    If Err.Number = 0 Then
      tstr = tstr & iProp.Title & D & iProp.Author & D & iProp.DateLastSaved
```

```
      Selection.TypeText Text:=tstr & vbCrLf
    Else
      tstr = tstr & Err.Description
      Selection.Font.Italic = True
      Selection.TypeText Text:=tstr & vbCrLf
      Selection.Font.Italic = False
    End If
    On Error GoTo 0
  Next i
  With Selection
    .WholeStory
    .Style = wdStyleNormal
    .Font.Size = 8
    .ConvertToTable Separator:=D, AutoFit:=True, Format:=wdTableFormatProfessional
    .Sort ExcludeHeader:=True, FieldNumber:=1
    .ParagraphFormat.SpaceBefore = 1
    .ParagraphFormat.SpaceAfter = 1
    .HomeKey Unit:=wdStory
    .SplitTable
    .Style = wdStyleHeading1
    .ParagraphFormat.SpaceAfter = 12
    .TypeText Text:="Iventar der Dokumente im Verzeichnis " & Pfad & vbCrLf
    .Style = wdStyleNormal
    .Delete Unit:=wdCharacter
    .HomeKey Unit:=wdStory
  End With
  Application.ScreenUpdating = True
End Sub
```

(Aus wenig Code entsteht ein Inventar aller Word-Dokumente im Standardverzeichnis.)

Der folgende VBA-Code listet alle Dokumentenvorlagen, welche sich im Standardvorlagenverzeichnis befinden, in einem Listenfeld, welches wiederum Bestandteile eines Dialoges ist (UserForm), auf. Beim Klick auf den Namen einer der Vorlagen wird deren Vorschaugrafik eingeblendet.

Kopieren Sie zuerst den Drei-Zeiler im VBA-Editor in ein neues Modul und fügen Sie dann eine **UserForm** ein.

Kopieren Sie nun den umfangreicheren Code in diese **UserForm**. Sie müssen keine Schaltflächen malen, da diese dynamisch erstellt werden. Achten Sie darauf, dass der Name der UserForm **UserForm1** lautet, wenn nicht, müssen Sie den Namen im Drei-Zeiler noch anpassen. Rufen Sie dann über den Word-Dialog ***Extras - Makro - Makros...*** das Makro mit dem Namen **VorschauGrafik** auf.

```
Sub VorschauGrafik()
  UserForm1.Show
End Sub
```

(Ruft die nun folgende UserForm auf.)

```
Private Const Titel = "Word-Vorlagen"
Private Const Text1 = "Klicken Sie auf eine Vorlage um die Vorschaugrafik zu sehen."
Private VG As MSForms.Image
Private Lab As MSForms.Label
Private WithEvents LB As MSForms.ListBox
Private WithEvents CB As MSForms.CommandButton
Private oFilePropReader As DSOleFile.PropertyReader
Private iProp As DSOleFile.DocumentProperties
Private Vorlage() As String

Private Sub UserForm_Initialize()
  With Me
    .Width = 228
    .Height = 234
    .Caption = Titel
  End With
  Set VG = Controls.Add("Forms.Image.1", "VG", True)
  With VG
    .BackColor = &H80000005
    .Top = 9
    .Left = 9
    .Width = 105
    .Height = 149
    .SpecialEffect = fmSpecialEffectEtched
    .PictureSizeMode = fmPictureSizeModeZoom
  End With
  Set LB = Controls.Add("Forms.ListBox.1", "LB", True)
  With LB
    .Top = 9
    .Left = 124
    .Width = 90
    .Height = 149
  End With
  Set CB = Controls.Add("Forms.CommandButton.1", "CB", True)
  With CB
    .Top = 165
    .Left = 124
    .Width = 90
    .Height = 18
    .Caption = "Schliessen"
    .Accelerator = "S"
  End With
  Set Lab = Controls.Add("Forms.Label.1", "Lab", True)
  With Lab
    .Top = 189
    .Left = 9
    .Width = 210
    .Height = 24
    .Caption = Text1
  End With
  Set oFilePropReader = New DSOleFile.PropertyReader
  With Application.FileSearch
    .NewSearch
    .FileType = msoFileTypeTemplates
```

```
    .LookIn = Options.DefaultFilePath(wdUserTemplatesPath)
    .SearchSubFolders = True
    .Execute SortBy:=msoSortByFileName
    If .FoundFiles.Count > 0 Then ReDim Vorlage(.FoundFiles.Count)
    j = -1
    For i = 1 To .FoundFiles.Count
      tmp = Dir(.FoundFiles(i))
      If tmp <> "" Then
        tmp = Left(tmp, Len(tmp) - 4)
        j = j + 1
        Vorlage(j) = .FoundFiles(i)
        LB.AddItem tmp
      End If
    Next i
  End With
  If j > -1 Then LB.ListIndex = 0
End Sub

Private Sub LB_Change()
  Dim Bild As StdPicture
  VG.Picture = Nothing
  V = Vorlage(LB.ListIndex)
  On Error Resume Next
  Set iProp = oFilePropReader.GetDocumentProperties(V)
  If Err.Number <> 0 Then
    Lab.Caption = Err.Description
    Err.Clear
    Exit Sub
  End If
  Set Bild = iProp.Thumbnail
  If Err.Number <> 0 Then
    If Err.Number = 424 Then
      Lab.Caption = "Die Vorlage enthält keine Vorschagrafik."
    Else
      Lab.Caption = Err.Description
    End If
    Err.Clear
    Exit Sub
  End If
  On Error GoTo 0
  VG.Picture = Bild
  Lab.Caption = Text1
End Sub

Private Sub CB_Click()
  Unload Me
End Sub

Private Sub UserForm_Deactivate()
  Set oFilePropReader = Nothing
  Set iProp = Nothing
End Sub
```

(Nun können Sie auch auf die Vorschaugrafik eines Office-Dokumentes zugreifen.)

Die Arbeit von David Foster

▲ Zurück zum Anfang

Dass Microsoft hier ein Mittel zur Verfügung stellt, ist relativ neu. Schon früher haben sich unabhängige Leute um diese Aufgabenstellung gekümmert. Dabei sticht vor allem die Arbeit von David Foster hervor, welche ebenfalls auf einer *ActiveX-DLL* basiert. Diese grossartige Arbeit besticht sowohl durch ihre Performanz beim Einlesen der Dateien, wie auch durch die einfache und kohärente Schnittstelle beim Aufruf ex VBA.

Sie brauchen dazu einzig eine DLL (Programmbibliothek) namens **FPropSet.dll**. Sie finden am Schluss dieses Skriptes den Link, welcher es Ihnen ermöglich diese DLL herunter zu laden.

Nachdem Sie die genannte DLL herunter geladen haben, sollten Sie diese auf Ihrem Rechner implementieren. Die Schritte dazu sind einfach und eigentlich immer erfolgreich. Gehen Sie wie folgt vor:

1. Kopieren Sie die Datei mit dem Namen **FPropSet.dll** in das Windows-Systemverzeichnis. Bei mir ist dies das Verzeichnis C:\Windows\System.
2. Als nächstes sollte Sie die DLL technisch registrieren. Wählen Sie dazu den Windows-Dialog ***Start - Ausführen***. In der Kommandozeile geben Sie folgendes ein: `regsvr32 FPropSet.dll`. Dieser Vorgang liefert eine Endmeldung (hoffentlich eine positive).
3. **Öffnen** Sie nun die Standardvorlage **Normal.dot** und wechseln Sie mit der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
4. Wählen Sie dort vom Hauptmenü den Dialog ***Extras - Verweise...***.
5. In der nun erscheinenden Liste suchen Sie nach dem Eintrag **FPropSet 1.0 Type Library** und aktivieren dort das Häkchen.
6. Speichern Sie nun die "Normal.dot" und verlassen Sie die Anwendung. Word sollte sich diese Einstellung für die Zukunft merken.

Die Syntax, welche dazu dient, die Eigenschaft einzulesen, wird Ihnen sicher von word-integrierten Objekten bekannt vorkommen. So wird's gemacht:

```
Sub IntegrierteEigenschaftLesen()
  Dim propset As FilePropertySet
  Set propset = New FilePropertySet
  propset.Pathname = "C:\Eigene Dateien\MeinDok.doc"
  Dim fps As FileProperties
  Set fps = propset.BuiltInFileProperties
  MsgBox fps("Author").Value
End Sub
```

(Die word-integrierte Dokumenteneigenschaft "Autor" wird ausgelesen, ohne das Dokument "MeinDok.doc" zu öffnen.)

```
Sub BenutzerdefinierteEigenschaftLesen()
  Dim propset As FilePropertySet
  Set propset = New FilePropertySet
  propset.Pathname = "C:\Eigene Dateien\MeinDok.doc"
  Dim fps As FileProperties
  Set fps = propset.CustomFileProperties
  MsgBox fps("Wohnort").Value
End Sub
```

(Die benutzerdefinierte Dokumenteneigenschaft "Wohnort" wird ausgelesen, ohne das Dokument "MeinDok.doc" zu öffnen.)

Folgendes Beispiel liest alle word-integrierten Dokumenteneigenschaft aus und stellt diese in einer Tabelle dar:

```
Sub AlleIntegriertenEigenschaftLesen()
  Set rDoc = Documents.Add
  Dim propset As FilePropertySet
  Set propset = New FilePropertySet
  propset.Pathname = "C:\Eigene Dateien\MeinDok.doc"
  Dim fps As FileProperties
  Dim fp As FileProperty
  Set fps = propset.BuiltInFileProperties
  Selection.TypeText Text:="Name" & vbTab & "Wert" & vbCrLf
  For Each fp In fps
    Selection.TypeText Text:=fp.Name & vbTab & fp.Value & vbCrLf
  Next
  With Selection
    .WholeStory
    .ConvertToTable Separator:=wdSeparateByTabs, AutoFit:=True, _
        Format:=wdTableFormatProfessional
    .HomeKey Unit:=wdStory
  End With
End Sub
```

(Sequentielles Lesen der word-integrierten Dokumenteneigenschaft, ohne das Dokument "MeinDok.doc" zu öffnen.)

Folgendes Beispiel liest einige word-integrierten Dokumenteneigenschaft für alle Dateien im nominierten Verzeichnis aus und stellt diese in einer Tabelle dar: Der Code ist in der abgedruckten Form lauffähig. Passen Sie allenfalls noch das Verzeichnis in der Variablen **Pfad** an.

Die Eigenschaften, welche ausgelesen werden, können Sie im Kopf des Codes bei Bedarf verändern. Setzen Sie zuerst in der Variable mit dem Namen **MaxProp** die Anzahl der Eigenschaften, welche Sie auslesen möchten. Anschliessend können Sie in den Variablen **Prop(1)** bis **Prop(n)** die **IDs** der gewünschten Eigenschaften setzen. Um herauszufinden, welche Eigenschaft welche ID trägt, verwenden Sie bitte die oben publizierte Tabelle.

```
Private MaxProp As Integer
Private Prop() As Long

Sub IntegrierteEigenschaftenEinesVerzeichnisesAuflisten2()
  Pfad = "c:\Eigene Dateien"
  Endung = ".doc"
  MaxProp = 4
  ReDim Prop(MaxProp)
  Prop(1) = 1  'Titel
  Prop(2) = 3  'Autor
  Prop(3) = 11 'Erstellt am:
  Prop(4) = 0  'Dateiname (Keine echte Dokumenteneigenschaft)
  Dim rDoc As Document
  Set rDoc = Documents.Add
  D = Chr(1)
  For i = 1 To MaxProp
    If Prop(i) = 0 Then
      tstr = tstr & "Dateiname"
    Else
```

```
      tstr = tstr & rDoc.BuiltInDocumentProperties(Prop(i)).Name
    End If
    If i < MaxProp Then tstr = tstr & D
  Next i
  Selection.TypeText Text:=tstr & vbCrLf
  chk = Dir(Pfad & "\*" & Endung)
  While chk <> ""
    EigenschaftenLesen Pfad, chk
    chk = Dir
  Wend
  With Selection
    .WholeStory
    .ConvertToTable Separator:=D, AutoFit:=True, Format:=wdTableFormatProfessional
    .Sort ExcludeHeader:=True, FieldNumber:=1
    .ParagraphFormat.SpaceBefore = 2
    .ParagraphFormat.SpaceAfter = 2
    .HomeKey Unit:=wdStory
    .SplitTable
    .Style = wdStyleHeading1
    .TypeText Text:="Iventar der Dokumente im Verzeichnis " & Pfad & vbCrLf
    .Style = wdStyleNormal
    .Delete Unit:=wdCharacter
    .HomeKey Unit:=wdStory
  End With
End Sub

Private Sub EigenschaftenLesen(Pfad, Datei)
  D = Chr(1)
  Dim propset As FilePropertySet
  Set propset = New FilePropertySet
  propset.Pathname = Pfad & "\" & Datei
  Dim fps As FileProperties
  Set fps = propset.BuiltInFileProperties
  For i = 1 To MaxProp
    If Prop(i) = 0 Then
      tstr = tstr & Datei
    Else
      On Error Resume Next
      tmp = fps(Prop(i))
      Fehler = Err.Number
      On Error GoTo 0
      If Fehler > 0 Then tmp = "---"
      If IsDate(tmp) Then tmp = Format(tmp, "dd/mm/yyyy hh:mm:ss")
      tstr = tstr & tmp
    End If
    If i < MaxProp Then tstr = tstr & D
  Next i
  Selection.TypeText Text:=tstr & vbCrLf
End Sub
```

(Liest die Dokumenteneigenschaften aller Dokumente im nominierten Verzeichnis aus und stellt diese in einer Tabelle dar.)

Links zu diesem Thema

▲ Zurück zum Anfang

Microsoft

► File: DSOFILE.EXE Lets You Read Document Poperties w/o Office Installed

David Foster

► FpropSet - Reading File Properties from Structured Storage Fiels

Eigene Links

► Informationen für die Word-Anwendung auch über einen Windows-Neustart hinaus sichern

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/DokEigen.htm

René Probst, Oktober 1999

Änderungen

- Dezember 1999 - DSOFILE (Microsoft hat es jetzt auch geschafft)

Vom Umgang mit Dokumentvariablen in MS Word (einschliesslich Tool zur Verwaltung von Dokumentvariablen)

◄ Zurück zum Hauptindex

Frage

> 1. Wie kann ich mit Word-VBA eine Dokumentvariable hinzufügen, den Wert setzen oder löschen?
>
> 2. MS Word bietet keine Benutzerschnittstelle, um Dokumentvariablen zu verwalten. Gibt es ein Tool, welches diese Verwaltung übernehmen kann?

Diese Seite

▼ Antwort
▼ Ziel und Zweck von Dokumentvariablen
▼ VBA-technischer Umgang mit Dokumentvariablen
▼ Den Wert einer Dokumentvariable im Dokument anzeigen
▼ Tool zur Verwaltung von Dokumentvariablen
▼ Links zu diesem Thema

Antwort

▲ Zurück zum Anfang

Im folgenden wir der VBA-technische Umgang mit Dokumentvariablen aufgezeigt. Anschliessend an diesen Teil, welcher sich an Leute mit Programmierkenntnisse richtet, finden Sie eine dialoggeführte Routine, welche es auf komfortable Art und Weise erlaubt, Dokumentvariablen ohne aktiven Einsatz von VBA zu administrieren.

Ziel und Zweck von Dokumentvariablen

▲ Zurück zum Anfang

Dokumentvariablen sind Elemente, welche in einem Dokument oder in einer Dokumentvorlage gespeichert sind. Beim Erstellen einer solchen Variablen vergeben Sie einen eindeutigen Namen unter der Sie später die Variable ansprechen können. Dieser Variablen weisen Sie dann einen Wert zu.

Grundsätzlich sehen Sie nichts von diesen Variablen im Dokument. Dank der Feldfunktion mit dem Namen **DokVariable** **Word 97** bzw. **DocVariable** **Word2000**, können Sie allerdings den Inhalt einer solchen Variablen an einer beliebiger Stelle in Ihrem Dokument anzeigen lassen.

Wird eine Dokumentvariable in einer Dokumentvorlage erstellt, dann vererbt sich diese Definition samt Inhalt an die Dokumente, welche auf Basis dieser Vorlage erstellt werden. Das selbe trifft auch auf die Feldfunktion DocVariable zu, mit welcher Sie einen Bezug auf eine solche Variable erstellen.

Dokumentvariablen sind dann sinnvoll, wenn Sie versteckte und adressierbare Informationen im Zusammenhang mit einem Word-Dokument über einen Neustart des Rechners hinaus sichern möchten.

Weitere Bemerkungen zu Dokumentvariablen

- Der **Name** einer Dokumentvariablen darf **maximal 255 Zeichen** lang sein. Die **Information**, welche sie in einer dieser Variablen speichern können, kann **fast beliebig lang** sein.
- MS Word bietet zwar eine integrierte Möglichkeit auf Dokumentvariablen zu verweisen. Eine integrierte Benutzerschnittstelle, um Dokumentvariablen zu erstellen bzw. zu löschen ist aber nicht vorgesehen. Das selbe trifft zu, um einer Dokumentvariable einen Wert zuzuweisen. Dieses Skript ist auch dazu da, dieses Leerraum zu füllen.

☛ MS Word bietet eine Alternative zu Dokumentvariablen, nämlich benutzerdefinierte Dokumenteigenschaften. Diese Eigenschaften sind eine Ergänzung zu den Word-integrierten Dokumenteigenschaften, wie z.B. Autor, Thema, usw. Mit diesen Dokumenteigenschaften können Sie ein sehr ähnliches Ziel verfolgen, wie mit Dokumentvariablen. Dokumenteigenschaften haben den Vorteil, dass Sie über den Dialog *Datei - Eigenschaften - Anpassen* **Word2000** über die Word-Oberfläche administriert werden können.

☹ Bedenken Sie auch, dass es unter **Word 97** (nicht aber in **Word2000**) ein sehr leidiges Problem im Zusammenhang mit Dokumentvariablen gibt. Wollen Sie nämlich eine Referenz (Feldfunktion) auf eine Dokumentvariable in das Dokument einfügen, geht das so lange gut, wenn sie diese Referenz im Hauptteil des Dokumentes einfügen. Tun Sie dies dagegen in einem Textfeld oder in der Kopf- bzw. Fusszeile, schmiert Ihnen Word reproduzierbar ab. Dieses Problem wird mit keinem der Office 97 Service Releases gelöst. Erst mit **Word2000** hat sich Microsoft erfolgreich bemüht, diesem unhaltbaren Zustand ein Ende zu setzen. Weichen Sie bei dieser Konstellation auf Dokumenteigenschaften aus.

VBA-technischer Umgang mit Dokumentvariablen

▲ Zurück zum Anfang

Dokumentvariable definieren (bekannt machen)

Im folgenden wird dem Dokument oder der Dokumentvorlage eine Dokumentvariable mit dem Namen *Wohnort* beigefügt.

```
ActiveDocument.Variables.Add Name:="Wohnort", Value:="Basel"
```

(Variable generieren und gleichzeitig einen Wert zuweisen.)

Anmerkung

- Beachten Sie, dass Sie gleichzeitig einen Wert zuweisen müssen. Dokumentvariablen ohne Inhalt sind nicht zulässig, werden diese doch eliminiert. Im Zweifel setzen Sie als Wert einen Leerschlag, wen der definitive Wert noch nicht bekannt ist.

Wert zuweisen

Im folgenden wird der Wert der Dokumentenvariablen mit dem Namen *Wohnort* neu zugewiesen.

```
ActiveDocument.Variables("Wohnort").Value = "Paris"
ActiveDocument.Fields.Update
```

(Wert neu zuweisen, also ändern und sicherstellen, dass der Feldwert im Dokument aktualisiert wird.)

- Beachten Sie, dass eine Leerzuweisung nicht möglich ist. Die Dokumentvariable würde dadurch gelöscht. Im Zweifel setzen Sie als Wert einen Leerschlag, wen der definitive Wert noch nicht bekannt ist.
- Wenn Ihr Dokument eine Referenz auf die Dokumentvariable beinhalten, müssen die Felder im Dokument, wie im Beispiel aktualisiert werden, damit der neu gesetzte Wert auch angezeigt wird. Sollte sich Ihre Referenz nicht im Hauptteil des Dokumentes, sondern in einem Textfeld bzw. in der Kopf- oder Fusszeile befinden (was faktisch nur in Word2000 möglich ist), dann lesen Sie das Kapitel: *Felder in allen Dokumentteilen aktualisieren*, etwas weiter unten in diesem Skript.

Dokumentvariable löschen

Im folgenden wird die Dokumentvariable mit dem Namen *Wohnort* gelöscht.

```
ActiveDocument.Variables("Wohnort").Delete
```

(Die Variable mit dem Namen "Wohnort" wird gelöscht)

Dokumentvariable abfragen

Im folgenden wird die Dokumentvariable mit dem Namen *Wohnort* abgefragt und sinnvoll darauf reagiert.

```
If ActiveDocument.Variables("Wohnort").Value = "Paris" _
   Then Sprache = "Französisch"
```

(Wert einer Dokumentenvariablen abfragen.)

Feststellen, ob eine bestimmte Dokumentvariable bereits im Dokument existiert

Leider bietet VBA keine direkte Möglichkeit, um festzustellen, ob eine gewisse Dokumentvariable bereits existiert. Sie müssten dies also ausprogrammieren. Im folgenden finden Sie zwei Code-Beispiele dazu, welche mit einer unterschiedlichen Mimik das selbe Ziel verfolgen.

```
Sub Test()
  Var = "Abteilung"
  If Not DocVarCheckExistence(Var) Then
    ActiveDocument.Variables.Add Name:=Var, Value:="RZ-Automation"
  End If
End Sub

Function DocVarCheckExistence(ByVal varName As String) As Boolean
  Dim oVar As Variable
  For Each oVar In ActiveDocument.Variables
    If oVar.Name = varName Then
      DocVarCheckExistence = True
      Exit Function
    End If
  Next
End Function
```

(Allgemeingültige Funktion, welche feststellt, ob eine Dokumentvariable mit einem bestimmten Namen bereits existiert.)

Eine Alternative ist der Frontalangriff unter gleichzeitigem Einsatz einer Fehlerroutine, wie dies im folgenden Beispiel gezeigt wird:

```
Sub DokVarDefinieren()
  On Error Resume Next
  ActiveDocument.Variables.Add Name:="Abteilung", Value:="RZ-Automation"
  rc = Err.Number
  msg = Err.Description
  On Error GoTo 0
  Select Case rc
    Case 0 'Frontalangriff erfolgreich.
    Case 5903 'Die DokVariable existierte bereits - Keine Aktion erforderlich.
    Case Else 'Unbekannter Fehler.
      MsgBox msg, vbCritical, "Meine Anwendung"
  End Select
End Sub
```

(Die Dokumentvariable wird ohne vorgängige Abfrage definiert. Die Fehlerroutine fängt einen allfälligen Fehler ab.)

Felder in allen Dokumentteilen aktualisieren

Angenommen Sie setzten den Wert einer Dokumentvariablen neu und Sie möchten diese Änderung im Dokument unverzüglich reflektieren, dann müssen Sie das entsprechende DocVariable-Feld im Dokument aktualisieren.

Sollte sich dieses Feld im Hauptteil des Dokumentes befinden, dann erreichen Sie dies sehr einfach, nämlich mit der VBA-Anweisung: **`ActiveDocument.Fields.Update`**

Sollte sich dieses Feld nun aber in einem Textfeld oder in der Kopf-/Fusszeile befinden, dann benötigen Sie dazu eine weiterreichende Mimik, nämlich die folgende:

```
ActiveDocument.Variables("Wohnort").Value = "Paris"
For Each Teil In ActiveDocument.StoryRanges
  Teil.Fields.Update
  While Not (Teil.NextStoryRange Is Nothing)
    Set Teil = Teil.NextStoryRange
    Teil.Fields.Update
  Wend
Next
```

(Wert einer Dokumentvariablen neu zuweisen, also ändern und sicherstellen, dass der Feldwert in allen Dokumentteilen aktualisiert wird.)

Die eben vorgestellten Mimiken, aktualisieren gleich alle Felder. Selbstverständlich können Sie allenfalls auch gezielter vorgehen. Die Grundproblematik, dass Sie mit dem Vorspann "ActiveDocument" ausschliesslich den Hauptteil des Dokumentes ansprechen, bleibt aber auch dann bestehen.

Den Wert einer Dokumentvariable im Dokument anzeigen

▲ Zurück zum Anfang

Angenommen, sie möchten den Wert der Dokumentvariablen mit dem Namen **Abteilung** an einer bestimmten Stelle im Dokument anzeigen lassen, dann gehen Sie wie folgt vor:

1. Setzen Sie im Dokument oder in der Dokumentvorlage die Einfügemarke an den Ort, an der die Abteilung erscheinen soll.
2. Vom Word-Hauptmenü wählen Sie nun den Dialog ***Einfügen - Feld...***.
3. In der linken Liste mit der Bezeichnung *Kategorien:* markieren sie den Eintrag **Dokumentautomation**.
4. In der rechten Liste mit der Bezeichnung *Feldnamen:* markieren Sie den Eintrag **DokVariable Word 97** oder **DocVariable Word2000**.
5. Als nächstes sollten Sie das Textfeld in diesem Dialog ergänzen. Setzen Sie also die Einfügemarke hinter den Feldnamen DokVariable (bzw. DocVariable); achten Sie darauf, dass eine Leerstelle vorhanden ist und tippen sie **"Abteilung"** und zwar inklusive Gänsefüsschen. Der ganze Ausdruck heisst nun also: **DokVariable "Abteilung"** bzw. **DocVariable "Abteilung"**.
6. Schliessen Sie nun den Dialog mit ***Ok*** ab und sichern Sie die Dokumentvorlage.

Den Wert einer Dokumentvariable im Dokument abfragen und darauf reagieren

Mit einer *WENN-Abfrage* **Word 97**, bzw. einer *IF-Abfrage* **Word2000** können Sie im Dokument den Wert einer bestimmten Dokumentvariable abfragen und darauf reagieren.

Angenommen Sie haben eine Dokumentvariable mit dem Namen **Sex**, welche entweder den Inhalt **f** für *weiblich* oder **m** für *männlich* als Wert haben kann, dann könnte diese Abfrage im Dokument wie folgt lauten:

```
{ WENN { DokVariable "Sex" } = "f" "Sehr geehrte Frau" "Sehr geehrter Herr" }
```

Word 97 bzw.

```
{ IF { DocVariable "Sex" } = "f" "Sehr geehrte Frau" "Sehr geehrter Herr" }
```

Word2000

Wie Sie eine Feldfunktion in ein Word Dokument eingeben können erfahren Sie gleich anschliessend.

Wie gibt man eine Feldfunktion in MS Word ein?

Im folgenden wird beschrieben, wie man eine Feldfunktion in ein Word-Dokument einbringt:

1. Betätigen Sie die Tastenkombination **[Alt + F9]** um die Feldcodes einzublenden.
2. Generieren Sie das Gerüst für den neuen Feldcode, indem Sie die Tastenkombination [**Strg + F9]** betätigen. Sie sollten nun ein Pärchen geschweifter Klammern sehen; die Einfügemarke steht mitten drin.
3. Setzen Sie nun die Einfügemarke zwischen dieses geschweifte Klammerpaar und tippen Sie die Funktion wie hier beschrieben ein.
4. Betätigen Sie nun wieder die Tastenkombination **[Alt + F9]** um den Feldcode auszublenden. Damit wird der Feldcode wieder ausgeblendet, an dessen Stelle erscheint nun das Ergebnis der Feldfunktion.
5. **Wichtig**: Nach jeder Änderung des Feldcodes sollten Sie die Taste **[F9]** betätigen, damit das Feld auch aktualisiert angezeigt wird. Dabei muss die Einfügemarke auf dem Feld stehen, bzw. das Feld muss markiert sein.

Tool zur Verwaltung von Dokumentvariablen

▲ Zurück zum Anfang

Im folgenden finden Sie ein sehr komfortables Tool um Dokumentvariablen zu verwalten. Somit sind keine VBA-Kenntnisse mehr von Nöten um:

Funktionalität

- Variablen zu definieren (festlegen).
- Den Wert von Variablen zu setzen.
- Variablen zu löschen.
- Variablen umzubenennen.
- Eine Liste aller Variablen eines Dokumentes oder in einer Dokumentenvorlage mit Inhalten zu erstellen.

Implementierung des Tools

Um das Tool verfügbar zu machen, gehen Sie bitte wie folgt vor:

1. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** (über ***Datei - Öffnen***).
2. Wechseln Sie mit der Tastenkombination **[Strg + F11]** in die VBA-Umgebung.
3. Wählen Sie im VBA-Editor im Hauptmenü den Dialog ***Einfügen - Modul*** und kopieren Sie den kurzen Code (**3-Zeiler**), welcher unten abgebildet ist, in das leere Code-Fenster.
4. Wählen Sie nun im Hauptmenü den Dialog ***Einfügen - UserForm***.
5. Führen Sie einen Rechtsklick auf die UserForm aus und wählen Sie im Kontextmenü den Punkt **Code anzeigen.**
6. Löschen Sie nun die zwei Code-Zeilen, welche von Word automatisch generiert wurden und kopieren Sie den grösseren Code, welcher unten abgebildet ist in das nun leere Fenster.
7. Lassen Sie sich nun mit der Taste **[F4]** den Eigenschaftsdialog der UserForm anzeigen.
8. Ändern Sie nun die Eigenschaft Name von *UserFormx* nach **frmDokVars**.
9. Kehren Sie in das Dokumentfenster zurück.
10. Wählen Sie dort vom Hauptmenü den Dialog ***Extras- Anpassen...***.
11. Aktivieren Sie den Reiter **Befehle**.
12. Markieren Sie in der linken Liste mit der Bezeichnung *Kategorien:* den Eintrag **Makros**.
13. Suchen Sie in der rechten Liste mit der Bezeichnung *Befehle* den Eintrag **DokVarsVerwalten**.
14. Ziehen Sie diesen Eintrag mit der Maus auf eine Symbolleiste Ihrer Wahl.
15. Optional können Sie immer noch in diesem Anpassungsmodus einen Rechtsklick auf die Schaltfläche ausüben, und über das Kontextmenü Name und Aussehen der Schaltfläche beeinflussen.
16. Speichern Sie nun die *Normal.dot* und verlassen Sie Word.
17. Nachdem Sie Word wieder gestartet habe, können Sie die neue Funktion nutzen.

Anmerkung

- Sie brauchen für die UserForm keine Schaltflächen zu zeichnen, da diese vom Programm dynamisch erstellt werden.

Zuerst also der kurze Code, welcher die UserForm aufruft. Anschliessend dann der Code, welchen Sie in die UserForm kopieren sollten.

VBA-Code

```
Sub DokVarsVerwalten()
  frmDokVars.Show
End Sub
```

(Ruft die UserForm mit dem Namen *DokVars* auf.)

```
'Dieser Word 97/2000-Dialog ermöglicht es Dokumentvariablen zu administrieren.
'Folgende Funktionen sind unterstützt:
'Erstellen, Wert ändern, Löschen, Umbenennen, Dump
'
'Erstellt am 10/12/00 von René Probst / eku / CH-8152 Glattbrugg
'
'Die Vervielfältigung (auch auf Datenträgern jeglicher Art), die
'Veröffentlichung und der Vertrieb mit kommerziellem Motiv ist
'untersagt!
'
Private WithEvents lbVars As MSForms.ListBox
Private txtName As MSForms.TextBox
Private txtWert As MSForms.TextBox
Private txtNeu As MSForms.TextBox
Private lblVars As MSForms.Label
Private lblStat As MSForms.Label
Private lblInhalt As MSForms.Label
Private lblNeu As MSForms.Label
Private WithEvents cmdNeu As MSForms.CommandButton
Private WithEvents cmdEdit As MSForms.CommandButton
Private WithEvents cmdEntfernen As MSForms.CommandButton
Private WithEvents cmdUmbenennen As MSForms.CommandButton
Private WithEvents cmdDump As MSForms.CommandButton
Private WithEvents cmdSchliessen As MSForms.CommandButton
Private AnzVars As Integer
Private Vars() As String
Private Const Titel = "Dokumentvariablen verwalten"

Private Sub UserForm_Initialize()
  Me.Caption = Titel
  Me.Height = 225
  Me.Width = 240
  DokVarsVerwaltenSchaltflaechenHinzufuegen
  DokVarsVerwaltenDialogAufbauen
  If AnzVars > 0 Then DokVarsVerwaltenVarsSammeln
End Sub

Private Sub DokVarsVerwaltenSchaltflaechenHinzufuegen()
  Set lblVars = Me.Controls.Add("Forms.Label.1", , True)
  With lblVars
    .top = 4
    .left = 9
```

```
  .Height = 12
  .Width = 150
End With
Set lblStat = Me.Controls.Add("Forms.Label.1", , False)
With lblStat
  .top = 102
  .left = 9
  .Height = 12
  .Width = 150
End With
Set lblInhalt = Me.Controls.Add("Forms.Label.1", , True)
With lblInhalt
  .top = 126
  .left = 9
  .Height = 12
  .Width = 222
End With
Set lblNeu = Me.Controls.Add("Forms.Label.1", , True)
With lblNeu
  .top = 42
  .left = 9
  .Height = 12
  .Width = 150
End With
Set txtWert = Me.Controls.Add("Forms.TextBox.1", "txtWert", True)
With txtWert
  .top = 138
  .left = 9
  .Height = 60
  .Width = 218
  .MaxLength = 2 ^ 16 - 1
  .MultiLine = True
  .EnterKeyBehavior = True
  .ScrollBars = fmScrollBarsVertical
End With
Set txtName = Me.Controls.Add("Forms.TextBox.1", "txtName", False)
With txtName
  .top = 18
  .left = 9
  .Height = 16
  .Width = 150
  .MaxLength = 255
End With
Set txtNeu = Me.Controls.Add("Forms.TextBox.1", "txtNeu", False)
With txtNeu
  .top = 57
  .left = 9
  .Height = 16
  .Width = 150
  .MaxLength = 255
End With
Set lbVars = Me.Controls.Add("Forms.ListBox.1", "lbVars", False)
With lbVars
  .top = 18
  .left = 9
  .Height = 90
  .Width = 150
End With
```

```
  oben = 9
  For i = 1 To 6
    Set cmd = Me.Controls.Add("Forms.CommandButton.1", , True)
    With cmd
      .top = oben
      .left = 168
      .Height = 18
      .Width = 60
      .Font.Size = 7.5
      Select Case i
        Case 1
          .Caption = "Neu..."
          .Accelerator = "N"
          Set cmdNeu = cmd
        Case 2
          .Caption = "Wert Ändern"
          .Accelerator = "Ä"
          Set cmdEdit = cmd
        Case 3
          .Caption = "Löschen"
          .Accelerator = "L"
          Set cmdEntfernen = cmd
        Case 4
          .Caption = "Umbenennen"
          .Accelerator = "U"
          Set cmdUmbenennen = cmd
        Case 5
          .Caption = "Dump"
          .Accelerator = "D"
          Set cmdDump = cmd
        Case 6
          .Caption = "Schliessen"
          .Accelerator = "S"
          .Cancel = True
          Set cmdSchliessen = cmd
      End Select
    End With
    oben = oben + 21
  Next i
End Sub

Private Sub DokVarsVerwaltenDialogAufbauen()
  AnzVars = ActiveDocument.Variables.Count
  If AnzVars = 0 Then
    DokVarsVerwaltenModus2
  Else
    DokVarsVerwaltenModus0
  End If
End Sub

Private Sub DokVarsVerwaltenVarsSammeln()
  lbVars.Clear
  i = -1
  ReDim Vars(AnzVars - 1)
  Dim oVar As Variable
  For Each oVar In ActiveDocument.Variables
    i = i + 1
    Vars(i) = oVar.Name
```

```
  Next
  If AnzVars > 1 Then WordBasic.SortArray Vars()
  lbVars.List = Vars()
  lbVars.ListIndex = 0
End Sub

Private Sub lbVars_Click()
  txtWert.Text = ActiveDocument.Variables(lbVars.Value).Value
  txtWert.SelStart = 0
End Sub

Private Sub cmdNeu_Click()
  If cmdNeu.Accelerator = "N" Then
    DokVarsVerwaltenModus2
    txtName.SetFocus
  Else
    strVar = CStr(Trim(txtName.Text))
    If strVar = "" Then
      MsgBox "Sie müssen einen Namen für die Variable festlegen.", vbInformation, Titel
      txtName.SetFocus
      Exit Sub
    End If
    If DokVarsVerwaltenCheckExistence(strVar) = True Then
      MsgBox "Diese Variable existiert bereits.", vbInformation, Titel
      txtName.SetFocus
      txtName.SelStart = 0
      txtName.SelLength = txtName.TextLength
      Exit Sub
    End If
    If txtWert.Text = "" Then
      MsgBox "Sie müssen einen Wert für die Variable festlegen.", vbInformation, Titel
      txtWert.SetFocus
      Exit Sub
    End If
    ActiveDocument.Variables.Add Name:=strVar, Value:=txtWert.Text
    DokVarsVerwaltenDialogAufbauen
    If AnzVars > 0 Then DokVarsVerwaltenVarsSammeln
    For i = 0 To lbVars.ListCount - 1
      If lbVars.List(i) = strVar Then
        lbVars.ListIndex = i
        Exit For
      End If
    Next
    txtWert.SetFocus
    txtWert.SelStart = 0
  End If
End Sub

Private Sub cmdEdit_Click()
  If DokVarsVerwaltenCheckExistence(lbVars.Value) = False Then
    MsgBox "Diese Variable wurde gelöscht.", vbInformation, Titel
    DokVarsVerwaltenDialogAufbauen
    If ActiveDocument.Variables.Count > 0 Then DokVarsVerwaltenVarsSammeln
    Exit Sub
  End If
  If txtWert.Text = "" Then
    MsgBox "Sie müssen einen Wert für die Variable festlegen.", vbInformation, Titel
    txtWert.Text = ActiveDocument.Variables(lbVars.Value).Value
```

```
    txtWert.SetFocus
    txtWert.SelStart = 0
    txtWert.SelLength = txtWert.TextLength
    Exit Sub
  End If
  ActiveDocument.Variables(lbVars.Value).Value = txtWert.Text
  txtWert.SetFocus
  txtWert.SelStart = 0
End Sub

Private Sub cmdEntfernen_Click()
  Q = Chr(34)
  If MsgBox("Solle die Variable mit dem Namen " & _
     Q & lbVars.Value & Q & " wirkilcht gelöscht werden?", _
        vbQuestion + vbOKCancel, Titel) = vbOK Then
    If DokVarsVerwaltenCheckExistence(lbVars.Value) = True Then
      ActiveDocument.Variables(lbVars.Value).Delete
      lbVars.RemoveItem lbVars.ListIndex
      AnzVars = AnzVars - 1
      If AnzVars = 0 Then DokVarsVerwaltenModus2
    Else
      DokVarsVerwaltenDialogAufbauen
      If AnzVars > 0 Then DokVarsVerwaltenVarsSammeln
    End If
  End If
End Sub

Private Sub cmdUmbenennen_Click()
  If cmdUmbenennen.Accelerator = "U" Then
    DokVarsVerwaltenModus1
    txtNeu.SetFocus
  Else
    strVar = CStr(Trim(txtNeu.Text))
    If strVar = "" Then
     MsgBox "Sie müssen den neuen Namen für die Variable festlegen.", vbInformation, Titel
      txtNeu.SetFocus
      Exit Sub
    End If
    If DokVarsVerwaltenCheckExistence(strVar) = True Then
      MsgBox "Die Variable mit diesem Namen existiert bereits.", vbInformation, Titel
      txtNeu.SetFocus
      txtNeu.SelStart = 0
      txtNeu.SelLength = txtNeu.TextLength
      Exit Sub
    End If
    ActiveDocument.Variables.Add Name:=strVar, Value:=txtWert.Text
    ActiveDocument.Variables(txtName.Text).Delete
    DokVarsVerwaltenDialogAufbauen
    If AnzVars > 0 Then DokVarsVerwaltenVarsSammeln
    For i = 0 To lbVars.ListCount - 1
      If lbVars.List(i) = strVar Then
        lbVars.ListIndex = i
        Exit For
      End If
    Next
    txtWert.SetFocus
    txtWert.SelStart = 0
  End If
```

```
End Sub

Private Sub cmdDump_Click()
  Dim nDoc As Document, oVar As Variable
  For Each oVar In ActiveDocument.Variables
    strText = strText & oVar.Name & Chr(1) & _
       DokVarsVerwaltenClean(CStr(oVar.Value)) & Chr(13)
  Next
  Unload Me
  DoEvents
  strName = ActiveDocument.FullName
  Set nDoc = Documents.Add
  nDoc.Content = "Name" & Chr(1) & "Inhalt" & vbCrLf & strText
  nDoc.Content.ConvertToTable Separator:=Chr(1), _
     Format:=wdTableFormatProfessional, AutoFit:=True
  nDoc.Content.Find.Execute FindText:=Chr(2), _
     ReplaceWith:="^p", Replace:=wdReplaceAll
  ActiveDocument.Range(0, 0).Select
  Selection.SplitTable
  Selection.Style = wdStyleHeading2
  Selection.TypeText "Dokumentvariablen von " & strName & " am " & _
     Format(Date, "dd/mm/yyyy") & " - " & Format(Time, "hh:mm:ss")
  ActiveDocument.Range(0, 0).Select
End Sub

Private Sub cmdSchliessen_Click()
  If ActiveControl.Accelerator = "S" Or AnzVars = 0 Then
    Unload Me
    Exit Sub
  End If
  DokVarsVerwaltenDialogAufbauen
  If lbVars.ListIndex > -1 Then
    txtWert.Text = ActiveDocument.Variables(lbVars.Value).Value
  End If
End Sub

Private Sub DokVarsVerwaltenModus0()
  lblVars.Caption = "Liste der Variablen in diesem Dokument:"
  txtName.Visible = False
  lblNeu.Visible = False
  txtNeu.Visible = False
  lbVars.Visible = True
  lblStat.Visible = True
  lblStat.Caption = "Das Dokument enthält " & AnzVars & " Variable(n)."
  lblInhalt.Visible = True
  lblInhalt.Caption = "Inhalt der markierten Variable:"
  cmdNeu.Caption = "Neu..."
  cmdNeu.Accelerator = "N"
  cmdUmbenennen.Caption = "Umbenennen"
  cmdUmbenennen.Accelerator = "U"
  cmdSchliessen.Caption = "Schliessen"
  cmdSchliessen.Accelerator = "S"
  cmdNeu.Enabled = True
  cmdEdit.Enabled = True
  cmdEntfernen.Enabled = True
  cmdUmbenennen.Enabled = True
  cmdDump.Enabled = True
End Sub
```

```
Private Sub DokVarsVerwaltenModus1()
  lblVars.Caption = "Aktueller Name:"
  lblNeu.Caption = "Neuer Name:"
  cmdUmbenennen.Caption = "Ok"
  cmdUmbenennen.Accelerator = "O"
  cmdUmbenennen.Default = True
  txtName.Visible = True
  txtName.Enabled = False
  lblNeu.Visible = True
  txtNeu.Visible = True
  txtName.Text = lbVars.Value
  txtNeu = ""
  txtNeu.SetFocus
  lbVars.Visible = False
  lblStat.Visible = False
  cmdSchliessen.Caption = "Abbrechen"
  cmdSchliessen.Accelerator = "A"
  cmdNeu.Enabled = False
  cmdEdit.Enabled = False
  cmdEntfernen.Enabled = False
  cmdDump.Enabled = False
  txtName.TabIndex = 0
  txtWert.TabIndex = 1
End Sub

Private Sub DokVarsVerwaltenModus2()
  lblVars.Caption = "Name der neuen Variable:"
  txtName.Visible = True
  txtName.Enabled = True
  lblNeu.Visible = False
  txtNeu.Visible = False
  txtName.Text = ""
  txtName.SetFocus
  lbVars.Visible = False
  lblStat.Visible = False
  txtWert.Text = ""
  lblInhalt.Caption = "Inhalt der neuen Variable eingeben:"
  cmdNeu.Enabled = True
  cmdNeu.Caption = "Festlegen"
  cmdNeu.Accelerator = "F"
  cmdNeu.Default = True
  cmdSchliessen.Caption = "Abbrechen"
  cmdSchliessen.Accelerator = "A"
  cmdEdit.Enabled = False
  cmdUmbenennen.Enabled = False
  cmdEntfernen.Enabled = False
  cmdDump.Enabled = False
  txtName.TabIndex = 0
  txtWert.TabIndex = 1
End Sub

Private Function DokVarsVerwaltenCheckExistence(ByVal Var As String) As Boolean
  Dim oVar As Variable
  For Each oVar In ActiveDocument.Variables
    If oVar.Name = Var Then
      DokVarsVerwaltenCheckExistence = True
      Exit For
```

```
    End If
  Next
End Function

Private Function DokVarsVerwaltenClean(strT As String) As String
  ofs = InStr(strT, Chr(13))
  While ofs > 0
    strT = left(strT, ofs - 1) & Chr(2) & _
       right(strT, Len(strT) - ofs - 1)
    ofs = InStr(strT, Chr(13))
  Wend
  DokVarsVerwaltenClean = strT
End Function
```

(Das ist der Code, welchen Sie in die UserForm mit dem Namen *frmDokVars* kopieren sollen. Die Schaltflächen werden vom Code dynamisch erstellt.)

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► Informationen für die Word-Anwendung auch über einen Windows-Neustart hinaus sichern

► Wie lese oder setze ich mit VBA den Wert einer Dokumenteneigenschaft?

Microsoft

► WD97: Invalid Page Fault Using DOCVARIABLE Field

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/DokVars.htm

René Probst, Dezember 2000

Änderungen

- --- nil ---

Wie kann ich bei zeitlich langandauernden VBA-Prozessen eine Fortschrittsleiste darstellen?

◄ Zurück zum Hauptindex

Problemstellung

> Ähnlich wie in der Statuszeile von Word, wenn z.B. ein Dokument gespeichert wird oder einen anderen längerer Prozess am laufen ist, möchte ich bei einem zeitintensiven VBA-Programm dem Benutzer einen Anhaltspunkt geben, dass das System nicht eingefroren ist, sondern dass der Prozess fortschreitet.

Vorausbemerkungen

- Eine Anzeige in der Statuszeile ist möglich und sicher auch sinnvoll. Allerdings müssten Sie Ihre Wünsche etwas abspecken. Grafik ist in der Statuszeile ex VBA **nicht** möglich.
- Die Fortschrittsleiste, welche man bei vielen Programmen auf einer Dialog-Box trifft, basiert auf der Programm-Datei **COMCTRL32.OCX**. Auf diese Datei dürfen Sie aber beim Programmieren nur zurückgreifen, wenn Sie neben Office auch Visual Basic 3/4/5/6 installiert haben. Sie dürfen auch nicht voraussetzen, dass diese Datei bei jedem Benutzer, welche Ihre Arbeit einsetzen möchte vorhanden ist, bzw. korrekt registriert ist. Aus den genannten Gründen, geht dieses Skript nicht weiter auf diesen Lösungsansatz ein.

Lösungsvarianten

Einfache Lösung mit Statusleiste

Sie könnten das Fortschreiten des Vorgangs mit einer Prozentzahl in der Status-Leiste indizieren. Es handelt sich dabei um eine Information, welche ausschliesslich aus reinem Text bestehen kann. Der dazu nötige VBA-Code ist unten abgebildet. Es ist leider nicht möglich, der Statuszeile ex VBA grafische Elemente zuzuweisen. Trotzdem sollte diese einfache Lösung in vielen Fällen genügen. Kopieren Sie nachfolgenden Code im VBA-Editor in ein neues Modul und testen Sie den Code. Sie sollten in der Statuszeile im normalen Word-Fenster beobachten können, wie die Prozentzahl hochgezählt wird. Gleichzeitig wird auch der Office Assistent animiert.

Fortschrittsanzeige in der Statuszeile...
...die Katze ist sehr, sehr beschäftigt.

```
Private Const Kopf = "Bis jetzt "
Private Const Schwanz = " beendet..."

Sub FortschritsanzeigeAufrufen()
  Assistant.Visible = True
  FortschrittAnzeigen 0, "Startet jetzt!"
  'Nun sollen z.B. 1000 Datensätze verarbeitet werden,...
  '...was hier mit einer sinnlosen Schleife simuliert wird.
  'Z.B. nach jedem 10 Datensatz wird die Funktion "Fortschrittsanzeige" aufgerufen.
  'Dabei übergeben Sie die Prozentzahl der bereits verarbeiteten Datensätze.
  AnzSätze = 1000
  Assistant.Animation = msoAnimationWorkingAtSomething
  For i = 1 To AnzSätze
    '**** Diese Schleife mit etwas sinnvollerem ersetzen
    For j = 1 To 100000 'Abgestimmt auf 400MHz-Prozessor
    Next j
    '**** Ende der Dummy-Schleife
    If i Mod 10 = 0 Then FortschrittAnzeigen i / AnzSätze * 100 ' Prozentzahl
  Next i
  FortschrittAnzeigen 100, "Jetzt beendet!"
  Assistant.Animation = msoAnimationCharacterSuccessMajor
End Sub

Public Function FortschrittAnzeigen(FS As Integer, Optional Text As String)
  If FS < 0 Or FS > 100 Then
    Err.Raise 32000, , "Prozentzahl ausserhalb des gültigen Bereiches."
  End If
  If Text = "" Then
    StatusBar = Kopf & FS & " %" & Schwanz
  Else
    StatusBar = Text
  End If
End Function
```

(Der Fortschritt des Prozesses wird in der Statusleiste dargestellt. Zusätzlich wird der Office Assistent animiert.)

Lösung mit einem Dialog (Modale UserForm in Office 97)

Die folgende Lösungsvariante benutzt eine **UserForm** um anzuzeigen, wie weit Ihr Prozess bereits fortgeschritten ist. Auch Ihr Code, welche den länger andauenden Prozess ausführt, sollte in dieser UserForm zu liegen kommen. Ersetzen Sie dazu einfach die Sinnlosschleife im folgenden Code durch Ihren eigenen Code.

Die Elemente auf der UserForm werden hier alle dynamisch erstellt, was bedeutet, dass Sie keine Steuerelemente zeichnen müssen.

Um dieses Beispiel zu testen, gehen Sie wie folgt vor:

1. Wechseln Sie mittels der Tastenkombination **[Alt + F11]** in die VBA-Umgebung.
2. Dort angekommen wählen Sie vom Hauptmenü den Dialog ***Einfügen - UserForm***.
3. Führen Sie einen Rechtsklick auf das Formular aus und wählen Sie im Kontextmenü ***Code anzeigen***.
4. **Löschen** Sie sämtlichen Code, welcher von Word automatisch generiert wurde.
5. Kopieren Sie den unten abgebildeten Code (der grössere von beiden) in das nun leere Fenster.
6. Vom Hauptmenü im VBA-Editor wählen Sie nun ***Einfügen - Modul***.
7. Kopieren Sie nun den unten abgebildeten Dreizeiler in dieses neue Fenster.
8. Kehren Sie in das Word-Fenster zurück und testen Sie den Code.

Anmerkung

Bei der Zeile `UserForm1.Show` muss **UserForm1** mit dem wirklichen Name der UserForm korrespondieren. Allenfalls müssen Sie dies anpassen. Wenn es die erste UserForm im Dokument ist, wählt Word allerdings immer den Namen "UserForm1".

```
Sub FortschrittsleisteFormularAufrufen()
  UserForm1.Show
End Sub
```

(Ruft die UserForm, welche die Fortschrittsleiste beherbergen auf.)

```
Private Const Titel = "Bitte warten..."
Private Const Kopf = "Bis jetzt "
Private Const Schwanz = " beendet..."
Private Const Abbruch = "Soll die Aktion wirklich abgebrochen werden?"
Private TB(25) As Control
Private Lab As Control

Private Sub UserForm_Initialize()
  Me.Caption = Titel
  Me.Width = 266
  Me.Height = 66
  tbLeft = 6
  For i = 1 To 25
    Set TB(i) = Me.Controls.Add("Forms.TextBox.1", "TB" & i, True)
    With TB(i)
      .Left = tbLeft
      .Top = 9
      .Width = 9
      .Height = 15
      tbLeft = tbLeft + 10
    End With
  Next i
```

```
  Set Lab = Me.Controls.Add("Forms.Label.1", "Lab1", True)
  With Lab
    .Left = 6
    .Top = 30
    .Width = 200
    .Height = 12
  End With
End Sub

Private Sub Userform_Activate()
  Report 0, "Startet jetzt!"
  'Nun sollen z.B. 1000 Datensätze verarbeitet werden,
  'was mit einer sinnlosen Schleife simuliert wird.
  'Nach jedem 10 Datensatz wird die Funktion "Report" aufgerufen.
  'Dabei übergeben Sie die Prozentzahl der bereits verarbeiteten Datensätze.
  AnzSätze = 1000
  For i = 1 To AnzSätze
    '**** Diese Schleife mit etwas sinnvollerem ersetzen
    For j = 1 To 100000 'Abgestimmt auf 400MHz-Prozessor
    Next j
    '**** Ende der Dummy-Schleife
    If i Mod 10 = 0 Then Report i / AnzSätze * 100 ' Prozentzahl übergeben
  Next i
  Report 100, "Jetzt beendet!"
  Unload Me
End Sub

Private Function Report(Fortschritt As Integer, Optional Text As String)
  If Fortschritt < 0 Or Fortschritt > 100 Then
    Err.Raise 32000, , "Prozentzahl ausserhalb des gültigen Bereiches."
  End If
  If Text = "" Then
    Lab.Caption = Kopf & Fortschritt & " %" & Schwanz
  Else
    Lab.Caption = Text
  End If
  For i = 1 To Fortschritt
    If i Mod 4 = 0 Then TB(i / 4).BackColor = &H80000002
  Next i
  DoEvents
End Function

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If CloseMode <> 1 Then
    Antwort = MsgBox(Abbruch, vbYesNo + vbCritical, Titel)
    If Antwort = 6 Then
      End
    Else
      Cancel = 1
  End If
End Sub
```

(Dies ist der Code für die UserForm. Der Code stellt eine Fortschrittsleiste dar und steuert diese auch. Am Schluss wird das Formular entladen. Es wird davon ausgegangen, dass Sie die darin enthaltene sinnlose Schleife durch Ihr Vorhaben ersetzen.)

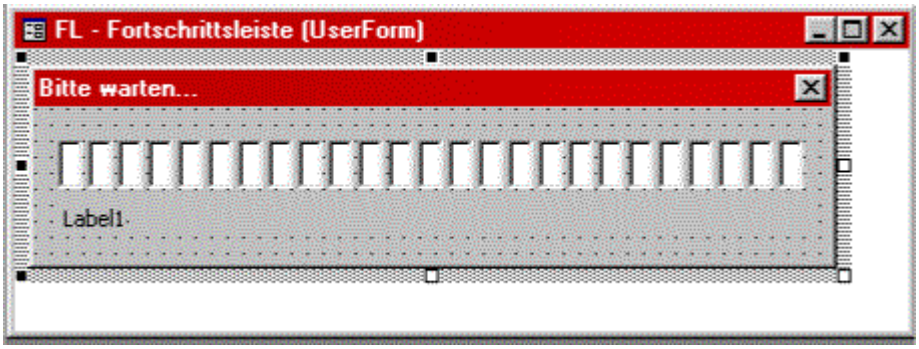
Lösung mit einem ungebundenen Dialog (nicht modale UserForm in Office2000)

Erst **Office2000** bietet uns die Möglichkeit, die Sache mit einem ungebundenen Dialog zu realisieren. Ein nicht modaler Dialog hat hier den Vorteil, dass unser Prozess trotz Aufruf der UserForm ziemlich ungehindert weiter läuft und nicht wie bei einem modalen Dialog wartet, bis die UserForm wieder Platz macht, also endet.

Nun können wir also aus dem vollen Schöpfen und uns eine echt modular aufgebaute Lösung erarbeiten. Der im Folgenden vorgeschlagene Weg, sieht eine einfach gestaltete UserForm vor und zusätzlich einen Prozess, welche die UserForm mit den aktuellen Daten (dem Fortschritt) treibt. Die Bausteine sind jetzt klar von einander abgegrenzt. Weder die UserForm noch der Prozess, welcher zwischen Ihrem langlaufenden Prozess und der UserForm liegt, müssen Sie, sowie die Sache mal eingerichtet ist, je wieder anpassen. Sie können diese beiden Prozesse quasi versiegeln.

Die UserForm

Für die UserForm schlage ich dieses Mal eine statische Lösung vor. Konkret heisst dies, Sie müssen diese UserForm selbst erstellen. Diese einfach gestaltete UserForm besteht aus den folgenden Teilen.



Die UserForm zur Entwurfszeit. Zeichen Sie 25 Textfelder und ein Bezeichnungsfeld

- Die **UserForm** selbst. Geben Sie dem Formular unter *Eigenschaften - Name* den Namen **Fortschrittsleiste**.
- Genau **25 Textfelder**. Nennen Sie diese Textfelder **TextBox1** bis **TextBox25**.
- Einem Bezeichnungsfeld. Nennen Sie dieses Bezeichnungsfeld **Label1**.

Als nächstes fügen Sie folgenden kurzen Code in die UserForm ein. Allfälliger Code, welcher Word automatisch generiert, können Sie löschen.

```
Private Const Titel = "Bitte warten..."
Private Const Kopf = "Bis jetzt "
Private Const Schwanz = " beendet..."
Private Const Abbruch = "Soll die Aktion wirklich abgebrochen werden?"

Private Sub UserForm_Initialize()
  Me.Caption = Titel
End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If CloseMode <> 1 Then
    Antwort = MsgBox(Abbruch, vbYesNo + vbCritical, Titel)
    If Antwort = 6 Then
      End
    Else
      Cancel = 1
    End If
  End If
End Sub
```

(Dies ist der Code für die nicht modale UserForm. Die darin enthaltenen Steuerelemente werden vom Treiber gesteuert.)

Der Treiber

Fügen Sie nun dem Projekt ein neues Modul bei und kopieren Sie den folgenden Code in dieses Modul:

```
Private Const Kopf = "Bis jetzt "
Private Const Schwanz = " beendet..."
Private TB(25) As Control

Function FortschrittsLeisteTreiber(ByVal Aktion As String, _
   Optional ByVal Fortschritt As Long, _
      Optional ByVal Text As String, _
         Optional ByVal Titel As String)
  Select Case Aktion
    Case "Start"
      FortschrittsLeiste.Show False
      Dim x As Control
      For Each x In FortschrittsLeiste.Controls
        If InStr(x.Name, "TextBox") = 1 Then
          n = Mid(x.Name, 8, 2)
          Set TB(n) = x
        End If
      Next
      If Titel = "" Then Titel = "Bitte warten..."
      FortschrittsLeiste.Caption = Titel
    Case "Ende"
      Unload FortschrittsLeiste
    Case "Anzeigen"
      If Not IsNumeric(Fortschritt) Then
        Err.Raise 32001, , "Prozentzahl nicht numerisch."
        End
      End If
      If Fortschritt < 0 Or Fortschritt > 100 Then
        Err.Raise 32002, , "Prozentzahl ausserhalb des gültigen Bereiches."
        End
      End If
    Case Else
      Err.Raise 32000, , "Ungüliger Wert für Aktion."
      End
  End Select
  If Text = "" Then
    FortschrittsLeiste.Label1.Caption = Kopf & Fortschritt & " %" & Schwanz
  Else
    FortschrittsLeiste.Label1.Caption = Text
  End If
  For i = 1 To Fortschritt
    If i Mod 4 = 0 Then TB(i / 4).BackColor = &H80000002
  Next i
  DoEvents
End Function
```

(Diese allgemeingültige Funktion treibt die UserForm.)

Der Klient

Schliesslich folgt noch der Code um das Werk zu testen. Einmal mehr führt der Code hier sinnlose Schleifen aus, um einen zeitintensiven Prozess zu simulieren. Periodisch oder auch spontan ruft der Prozess den Treiber auf, damit die Fortschrittsleiste aktualisiert werden kann. Verwenden Sie beim Aufruf des Treibers folgende Parameters:

...und die Fortschritssleiste während der Laufzeit.

- Der erste Parameter ist zwingend und beschreibt die Aktion. Sie müssen hier einen der folgenden Werte angeben:
  - **Start** - Damit wird der Dialog angezeigt. Optional können Sie als vierten Parameter noch den Titel des Dialoges übergeben.
  - **Anzeigen** - Damit wird der Dialog aktualisiert. Sie müssen zwingend als zweiter Parameter eine Prozentzahl zwischen 1 und 100 übergeben.
  - **Ende** - Sorgt dafür, dass die UserForm endet. Weitere Parameters sind hier nicht vorgesehen.
- Der zweite Parameter beschreibt den Fortschritt. Übergeben Sie eine Zahl zwischen 1 und 100.
- Der dritte Parameter ist optional. Er kann dafür benutzt werden, einen Text zu übergeben, welcher an Stelle der Prozentzahl angezeigt werden soll, also zum Beispiel: *Die Installation wird jetzt initialisiert*.
- Der vierte Parameter kann dazu benutzt werden, den Standardtext im Titel des Dialoges zu überschreiben. Der Standardtitel lautet *Bitte warten...*.

```
Sub MeinZeitintensiverProzess()
  FortschrittsLeisteTreiber "Start", , , "Ich bin an der Arbeit..."
  For i = i To 100
    For j = 1 To 500000
    Next j
    FortschrittsLeisteTreiber "Anzeigen", i
  Next i
  FortschrittsLeisteTreiber "Ende"
End Sub
```

(Diese sinnlose Schleife testet unsere Fortschrittsleiste.)

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/FL.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen?
Wie kann ich eine Liste der Word-Befehle erstellen?

◄ Zurück zum Hauptindex

Frage

> 1. Ich schreibe immer mal wieder benutzerdefinierte Funktionen in Word. Um diese Funktionen zugänglich zu machen, lege ich einen Schaltknopf auf einer der Symbolleisten an und verknüpfe diesen Schalter mit dem Makro. Natürlich möchte ich diesen Schalter mit einem Sinnbild kennzeichnen. Wenn ich den Schalter ex VBA generiere, brauche ich dazu aber die sogenannte *FaceID*. Wie komme ich zu einer Liste, welche mir alle in Word zur Verfügung stehenden Sinnbilder anzeigt und auch die dazugehörige *FaceID*?
>
> 2. Wie komme ich zu einer übersichtlichen Liste. aller Wordbefehle?

Antwort

Hier ein VBA-Programm, welches folgendes bewerkstelligt: Das Programm kann...

- ☺ ...sämtliche Symbolleisten ermitteln und tabellarisch darstellen. Dabei wird sowohl der interne (englischer) Name sowie der deutsche Name der Leiste aufgelistet.
- ☺ ...fast alle Word-Befehle aufspüren und diese mit Sinnbild (wo vorhanden) und Bedeutung (Beschreibung) des Befehles in einer Tabelle darstellen.
- ☺ Fast alle Word-Symbole (auch Sinnbilder oder Ikonen genannt) mit ihrer **FaceID** (das ist die interne und eindeutige Nummer jedes einzelnen Sinnbildes) anzeigen.
- ☺ Ein Word-Dokument mit Inhaltsverzeichnis erzeugen, welches alle drei Funktionen, wie oben beschriebenen, umfasst.

Die Funktion ist verwendet zur Benutzerführung den Office Assistenten. Der Ablauf ist dann eigentlich selbsterklärend.

Kopieren Sie den gesamten Code im VBA-Editor in ein neues Modul und rufen Sie unter ***Extras - Makro - Makros...*** die Prozedur mit dem Namen **SymbolleistenInventur** auf.

Vermeiden Sie es bitte, während den paar Minuten, in denen das Programm abläuft, das Word-Fenster zu wechseln. Nachdem die Prozedur abgelaufen ist, können Sie das erstellte Dokument abspeichern, damit es Ihnen immer zur Verfügung steht.

Achtung: Die Routine funktioniert z.Z. ausschliesslich in **Word2000**.

VBA-Code

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
' Dieses Modul enthält vier Routinen:
' Die Routine "SymbolleistenTabelle" erstellt eine Liste aller Symbolleisten.
' Die Routine "SymbolleistenBefehlsTabelle" erstellt eine Liste aller Befehle.
' Die Routine "SymbolleistenFaceIDTabelle" erstellt eine Liste aller Sinnbilder.
' Die Routine "SymbolleistenReport" ruft alle oben erwähnten Routinen auf.
' Alle Reports werden dabei in einem Dokument zusammengefasst, am Anfang des
' Reports wird ein Index geschrieben.
'
' Erstellt am 20/11/99 von René Probst / eku / CH-8152 Glattbrugg
'
'Effektiv ist der höchste Befehl für Word 97 3517 und für Word2000 4387
'Allerdings gibt es in den oberen Regionen kaum noch Treffer.
'Zudem schmiert Word ab; irgendwo bei 3512 beim Start bei 1. (It's Microsoft!)
Private Const RealBefehl = 3500
Private Const RealFaceID = 3500
Private Const TestBefehl = 50
Private Const TestFaceID = 150
Private Const StartBefehl = 1
Private Const StartFaceID = 1
Private Const MaxCol = 10
'SaveInterval nicht zu hoch ansetzen, sonst schmiert Word ab (It's Microsoft!)
Private Const SaveInterval = 250
Private Const asTitle = "Inventur der Symbolleisten, Word-Befehle und Sinnbilder"
Private Modus As Integer
Private Inhalt() As String
Private tmp() As String
Private tmpZähler As Integer
Private asVisible As Boolean
Private asText As String
Private asInventur As Boolean
Private asSpelling As Boolean
Private asGrammer As Boolean
Private asPlaceHolders As Boolean
Private asCreateBackground As Boolean
Private asAllowFastSave As Boolean
Private asBackgroundSave As Boolean
Private asSaveInterval As Long
Private oBlase As Balloon
Private oTable As Table
Private ComBar As CommandBar
Private x As CommandBarControl
Private wDoc As Document
Public rDoc As Document

Sub SymbolleistenInventur()
  asVisible = Assistant.Visible
  asInventur = False
  ReDim Inhalt(3, 1)
```

```
  Inhalt(0, 0) = "Inhalt"
  Inhalt(0, 1) = "Inhalt"
  Inhalt(1, 0) = "Symbolleisten"
  Inhalt(1, 1) = "Liste der Symbolleisten"
  Inhalt(2, 0) = "Befehle"
  Inhalt(2, 1) = "Liste der Befehle"
  Inhalt(3, 0) = "Icons"
  Inhalt(3, 1) = "Liste der Sinnbilder (FaceID's)"
  Assistant.Visible = True
  Wahl = BlaseAnzeigen
  Modus = 1
  ReDim tmp(2)
  tmpZähler = 0
  Select Case Wahl
    Case 1
      SymbolleistenTabelle
    Case 2
      SymbolleistenBefehlsTabelle
    Case 3
      SymbolleistenFaceIDTabelle
    Case 4
      SymbolleistenReport
    Case 5
      Modus = 0
      SymbolleistenReport
    Case -2
  End Select
  HappyEnd
End Sub

Private Sub SymbolleistenReport()
  asInventur = True
  SymbolleistenTabelle
  rDoc.Windows(1).Visible = True
  rDoc.Activate
  With Selection
    .MoveUp
    .Style = wdStyleHeading1
    .Font.Size = 15
    .TypeText Text:="Symbolleistenreport"
    .ParagraphFormat.SpaceAfter = 6
    .ParagraphFormat.Alignment = wdAlignParagraphCenter
    .TypeParagraph
    .Style = wdStyleHeading2
    .Bookmarks.Add Name:=Inhalt(0, 0), Range:=Selection.Range
    .Font.Size = 13
    .TypeText Text:=Inhalt(0, 1)
    .ParagraphFormat.SpaceAfter = 3
    .ParagraphFormat.Alignment = wdAlignParagraphCenter
    .TypeParagraph
    For i = 1 To 3
      .TypeText Text:=Inhalt(i, 1)
      .HomeKey Unit:=wdLine, Extend:=1
```

```
      .Hyperlinks.Add Anchor:=Selection.Range, SubAddress:=Inhalt(i, 0)
      .HomeKey Unit:=wdLine, Extend:=1
      .Font.Bold = True
      .ParagraphFormat.Alignment = wdAlignParagraphCenter
      .EndKey Unit:=wdLine
      .TypeParagraph
    Next i
    .TypeParagraph
    .ParagraphFormat.Alignment = wdAlignParagraphLeft
    .InsertBreak Type:=wdSectionBreakNextPage
    .Delete Unit:=wdCharacter
    .HomeKey Unit:=wdStory
  End With
  ActiveDocument.Sections(1).PageSetup.VerticalAlignment = wdAlignVerticalCenter
  rDoc.Windows(1).Visible = False
  SymbolleistenBefehlsTabelle
  SymbolleistenFaceIDTabelle
  StatusBar = "Listings beendet."
  asInventur = False
  HappyEnd
End Sub

Private Sub SymbolleistenTabelle()
  Reporting 0, "Tabelle der Symbolleisten wird vorbereitet."
  SymbolleistenStart
  Selection.TypeText Text:="Interner Name" + vbTab + "Benutzername" + vbTab + _
     "Typ" + vbTab + "Integriert" + vbTab + "ID" + vbCrLf
  Dim Leiste As CommandBar
  i = 0
  For Each Leiste In Application.CommandBars
    If Leiste.Name <> "*Test*" Then
      i = i + 1
      Reporting i, "Symbolleisten verarbeitet"
      Select Case Leiste.BuiltIn
        Case 0: Integriert = "Nein"
        Case 1: Integriert = "Ja"
      End Select
      Select Case Leiste.Type
        Case 0: Typ = "Normal"
        Case 1: Typ = "MenuBar"
        Case 2: Typ = "PopUp"
      End Select
      Selection.TypeText Text:=Leiste.Name + vbTab + Leiste.NameLocal + _
         vbTab + Typ + vbTab + Integriert + vbTab + CStr(Leiste.Index) + vbCrLf
    End If
  Next
  TabelleErstellen 1
  TabelleFormatieren Inhalt(1, 0), Inhalt(1, 1)
  Reporting 0, "Symbolleistentabelle erstellt."
End Sub

Private Sub SymbolleistenBefehlsTabelle()
  If Modus = 0 Then MaxBefehl = TestBefehl Else MaxBefehl = RealBefehl
```

```
  Reporting 0, "Tabelle der Befehle wird vorbereitet."
  SymbolleistenStart
  Text = "ID" & vbTab & "Befehl" & vbTab & "Icon" & vbTab & _
     "Beschreibung" & vbCrLf
  Selection.TypeText Text:=Text
  j = 1
  If StartBefehl < 2 Then c = 2 Else c = StartBefehl
  For i = c To MaxBefehl
    j = j + 1
    On Error Resume Next
    Set x = ComBar.Controls.Add(ID:=i)
    Fehler = Err.Number
    On Error GoTo 0
    If Fehler = 0 Then
      bName = x.Caption
      ofs = InStr(bName, "&")
      If ofs > 0 Then
        If ofs = 1 Then
          bName = Right(bName, Len(bName) - 1)
        Else
          bName = Left(bName, ofs - 1) + Right(bName, Len(bName) - ofs)
        End If
      End If
      Selection.TypeText Text:=i & vbTab & bName & vbTab
      If x.Type = msoControlButton Then
        x.CopyFace
        Selection.Paste
      End If
      Selection.TypeText Text:=vbTab & x.DescriptionText & vbCrLf
      Reporting j, "Befehle verarbeitet."
      If j Mod SaveInterval = 0 Then wDoc.Save
    End If
  Next i
  wDoc.Save
  TabelleErstellen 2
  TabelleFormatieren Inhalt(2, 0), Inhalt(2, 1)
  Reporting 0, "Tabelle der Befehle erstellt."
End Sub

Private Sub SymbolleistenFaceIDTabelle()
  If Modus = 0 Then MaxFaceId = TestFaceID Else MaxFaceId = RealFaceID
  Reporting 0, "Tabelle der Sinnbilder (Face-ID's) wird vorbereitet."
  SymbolleistenStart
  For i = 1 To MaxCol
    strText = strText + "ID" + vbTab + "Icon" + vbTab
  Next i
  Selection.TypeText Text:=Left(strText, Len(strText) - 1) + vbCrLf
  Set x = ComBar.Controls.Add(ID:=1)
  For i = StartFaceID To MaxFaceId
    Selection.TypeText Text:=i & vbTab
    On Error Resume Next
    x.FaceId = i
    x.CopyFace
```

```
    Selection.Paste
    If i Mod MaxCol = 0 Then
      Selection.TypeParagraph
    Else
      Selection.TypeText Text:=vbTab
    End If
    Reporting i, "Sinnbilder verarbeitet."
    If i Mod SaveInterval = 0 Then wDoc.Save
  Next
  TabelleErstellen 3
  TabelleFormatieren Inhalt(3, 0), Inhalt(3, 1)
  Reporting 0, "Tabelle der Sinnbilder (Face-ID's) erstellt."
End Sub

Private Sub TabelleErstellen(tCode)
  wDoc.Save
  Text = "Die Tabelle wird jetzt erstellt. "
  Reporting 0, Text & "- Das kann mehrere Minuten dauern."
  Assistant.Animation = msoAnimationSearching
  DoEvents
  Selection.WholeStory
  Select Case tCode
    Case 1
      Selection.ConvertToTable Separator:=wdSeparateByTabs, _
         Format:=wdTableFormatProfessional, AutoFit:=True
      Set oTable = wDoc.Tables(1)
    Case 2
      Selection.ConvertToTable Separator:=wdSeparateByTabs, _
         Format:=wdTableFormatProfessional, AutoFit:=True
      Set oTable = wDoc.Tables(1)
      oTable.Columns(1).Select
      Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
    Case 3
      Selection.ConvertToTable Separator:=wdSeparateByTabs, _
         Format:=wdTableFormatProfessional, AutoFit:=True
      Set oTable = wDoc.Tables(1)
      For i = 1 To oTable.Columns.Count Step 2
        oTable.Columns(i).Select
        Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
      Next i
  End Select
End Sub

Private Sub TabelleFormatieren(tSubjekt, tTitel)
  wDoc.Save
  Text = "Die Tabelle wird jetzt formatiert. "
  Reporting 0, Text & "- Das wird nochmals eine Weile dauern."
  Assistant.Animation = msoAnimationGetWizardy
  DoEvents
  If oBlase.Checkboxes(1).Checked = True And tSubjekt <> "Icons" Then
    oTable.Sort ExcludeHeader:=True, FieldNumber:=2
  End If
  oTable.Rows(1).HeadingFormat = True
```

```
  oTable.Rows.AllowBreakAcrossPages = False
  If tSubjekt = "Befehle" Then
    oTable.Columns(2).Width = CentimetersToPoints(3.5)
    oTable.Columns(3).Width = CentimetersToPoints(1)
    oTable.Columns(4).Width = CentimetersToPoints(12)
  End If
  rDoc.Activate
  With Selection
    .EndKey Unit:=wdStory
    If rDoc.Tables.Count > 0 Then .InsertBreak Type:=wdPageBreak
    .Bookmarks.Add Name:=tSubjekt, Range:=Selection.Range
    .Style = wdStyleHeading2
    .TypeText Text:=tTitel
    .ParagraphFormat.SpaceAfter = 15
    .TypeParagraph
    If asInventur = True And rDoc.Tables.Count > 0 Then
      EinfügenVerzeichnis tSubjekt
    End If
    .Style = "Standard"
    Assistant.Animation = msoAnimationGetTechy
    .FormattedText = wDoc.Content.FormattedText
    wDoc.Close SaveChanges:=wdDoNotSaveChanges
    .GoTo What:=wdGoToBookmark, Name:=tSubjekt
  End With
End Sub

Private Sub EinfügenVerzeichnis(tSubjekt)
  rDoc.Windows(1).Visible = True
  With Selection
    For i = 0 To 3
      If Inhalt(i, 0) <> tSubjekt Then
        .TypeText Text:=Inhalt(i, 1)
        .HomeKey Unit:=wdLine, Extend:=1
        .Hyperlinks.Add Anchor:=Selection.Range, SubAddress:=Inhalt(i, 0)
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
        .TypeParagraph
      End If
    Next i
    .TypeParagraph
    .ParagraphFormat.Alignment = wdAlignParagraphLeft
  End With
End Sub

Private Sub SymbolleistenStart()
  System.Cursor = wdCursorWait
  DoEvents
  On Error Resume Next
  rDoc.Activate
  If Err.Number > 0 Then
    On Error GoTo 0
    Set rDoc = Documents.Add
    rDoc.SaveAs FileName:="Symbolleistenreport vom " + _
       Format(Date, "dd-mm-yy") + " " + Format(Time, "hhmmss")
```

```
    SeiteEinrichten
    FusszeileEinrichten
    If asInventur = True Then Selection.TypeParagraph
  Else
    rDoc.Save
  End If
  On Error GoTo 0
  rDoc.Windows(1).Visible = False
  Set wDoc = Documents.Add
  tmp(tmpZähler) = CurDir & "\" & "tmp" & Format(Time, "hhmmss") + ".doc"
  wDoc.SaveAs FileName:=tmp(tmpZähler)
  tmpZähler = tmpZähler + 1
  SeiteEinrichten
  Application.ScreenUpdating = False
  asPlaceHolders = ActiveWindow.View.ShowPicturePlaceHolders
  With ActiveWindow.View
    .Type = wdPageView
    .ShowPicturePlaceHolders = True
  End With
  With Options
    asSpelling = .CheckSpellingAsYouType
    asGrammer = .CheckGrammarAsYouType
    asCreateBackground = .CreateBackup
    asAllowFastSave = .AllowFastSave
    asBackgroundSave = .BackgroundSave
    asSaveInterval = .SaveInterval
    .CreateBackup = False
    .CheckSpellingAsYouType = False
    .CheckGrammarAsYouType = False
    .AllowFastSave = False
    .BackgroundSave = False
    .SaveInterval = 0
  End With
  CustomizationContext = wDoc
  Set ComBar = CommandBars.Add(Name:="*tmp" & Format(Time, "hhmmss"))
  Assistant.Animation = msoAnimationWorkingAtSomething
End Sub

Private Sub SeiteEinrichten()
  With ActiveDocument.PageSetup
    .Orientation = wdOrientPortrait
    .TopMargin = CentimetersToPoints(0.9)
    .BottomMargin = CentimetersToPoints(2.5)
    .LeftMargin = CentimetersToPoints(2.3)
    .RightMargin = CentimetersToPoints(1.2)
    .FooterDistance = CentimetersToPoints(1.8)
  End With
  With Selection
    .Style = "Standard"
    .Font.Size = 8
  End With
  With ActiveDocument.Styles("Standard").ParagraphFormat
    .SpaceBefore = 2
```

```
    .SpaceAfter = 1
    .LineSpacingRule = wdLineSpaceAtLeast
    .LineSpacing = 9
  End With
End Sub

Private Sub FusszeileEinrichten()
  Q = Chr(34)
  rDoc.Sections(1).Footers(1).Range.Select
  With Selection
    .Fields.Add Range:=Selection.Range, Type:=wdFieldStyleRef, _
       Text:=Q & rDoc.Styles(wdStyleHeading2).NameLocal & Q
    .EndKey
    .TypeText Text:=vbTab
    .InsertAfter "Seite "
    .EndKey
    .Fields.Add Range:=Selection.Range, Type:=wdFieldPage
    .EndKey
    .InsertAfter " von "
    .EndKey
    .Fields.Add Range:=Selection.Range, Type:=wdFieldNumPages
  End With
  With Selection.ParagraphFormat
    .TabStops.ClearAll
    .Alignment = wdAlignParagraphRight
     pos = Selection.Information(wdHorizontalPositionRelativeToPage)
    .Alignment = wdAlignParagraphLeft
    .TabStops.Add Position:=pos, Alignment:=wdAlignTabRight
  End With
  ActiveWindow.ActivePane.Close
End Sub

Private Sub Reporting(tCount, tText)
  If tCount > 0 Then tText = CStr(tCount) + " " + tText
  StatusBar = tText
  DoEvents
  On Error Resume Next
  wDoc.UndoClear
  rDoc.UndoClear
  On Error GoTo 0
End Sub

Private Sub HappyEnd()
  Assistant.Animation = msoAnimationCharacterSuccessMajor
  ActiveWindow.View.ShowFieldCodes = False
  If asIventur = False Then
    ActiveWindow.View.ShowPicturePlaceHolders = asPlaceHolders
    Selection.HomeKey Unit:=wdStory
    Assistant.Visible = asVisible
    System.Cursor = wdCursorNormal
    Application.ScreenUpdating = True
  End If
  On Error Resume Next
```

```
  rDoc.Activate
  If Err.Number = 0 Then
    On Error GoTo 0
    rDoc.Content.LanguageID = wdNoProofing
    rDoc.Save
    rDoc.Activate
    rDoc.Windows(1).Visible = True
  End If
  On Error GoTo 0
  With Options
    .CheckSpellingAsYouType = asSpelling
    .CheckGrammarAsYouType = asGrammer
    .CreateBackup = asCreateBackground
    .AllowFastSave = asAllowFastSave
    .BackgroundSave = asBackgroundSave
    .SaveInterval = asSaveInterval
  End With
  Assistant.Animation = msoAnimationGetArtsy
  Set wDoc = Nothing
  For i = 0 To 2
    On Error Resume Next
    If tmp(i) <> "" Then Kill tmp(i)
    On Error GoTo 0
  Next i
End Sub

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = asTitle
    asText = "Diese Routine erstellt Listen für die in Word integrierten "
    asText = asText & "Symbolleisten, Befehle und Sinnbilder für die "
    asText = asText & "Schaltflächen." & vbCrLf & vbCrLf + "Bitte beachten Sie, "
    asText = asText & "dass die meisten dieser Routinen eine längere Laufzeit "
    asText = asText & "aufweisen." & vbCrLf & "Die unten angegebene Zeitdauer "
    asText = asText & " ist als Richtwert gedacht und bezieht sich auf einen "
    asText = asText & "Prozessor, welcher mit 200 MHz getaktet wird."
    .Text = asText + vbCrLf + vbCrLf + "Was möchten Sie tun ?"
    .Labels(1).Text = "Eine Liste aller Symbolleisten erstellen." + vbCrLf + _
       "(Dauer: < 1 Min.)"
    .Labels(2).Text = "Eine Liste aller Word-Befehle erstellen." + vbCrLf + _
        "(Dauer: ca. 3 Min.)"
    .Labels(3).Text = "Eine Liste aller Sinnbilder erstellen." + vbCrLf + _
       "(Dauer: ca. 6 Min.)"
    .Labels(4).Text = "Alle oben genannten Listen erstellen." + vbCrLf + _
        "(Dauer: ca. 8 Min.)"
    temp = "Alle genannten Listen erstellen aber nur mit einigen "
    .Labels(5).Text = temp & "Elementen zum Testen." + vbCrLf + _
        "(Dauer: ca. 1 Min.)"
    .Checkboxes(1).Text = "Alphabetische Sortierreihenfolge"
```

```
    .Button = msoButtonSetCancel
    BlaseAnzeigen = .Show
  End With
End Function
```

(Erstellt ein Inventar der Symbolleisten, Befehle und Sinnbilder in Word.)

Anmerkungen zum Code

! Die wenigen Befehle und Sinnbilder (ca. 3-4%), deren ID höher als 3500 ist, werden bewusst nicht dargestellt, da Word auf Grund einer merkwürdigen Konstellation (eine Mischung des Volumens und von korrupten Sinnbilder) reproduzierbar abschmiert, falls auch die Befehle und Sinnbilder in den oberen Regionen der Skala dargestellt werden. Dieses Problem lässt sich sowohl in Word 97 wie auch in Word2000 reproduzieren, wobei die Fehlermeldung in Word2000 anders ist, als in Word 97. Es liegt weder an der Grafikkarte und deren Treiber noch sonst an der Rechnerumgebung: It's just Microsoft! In der hier publizierten Version, ist das Programm aber hinreichend getestet und dürfte zu keinen Problemen führen.

☛ Wenn Sie das Makro starten, werden Sie auch eine Option entdecken, welche es Ihnen erlaubt, das Programm mit wenigen Befehlen bzw. Sinnbildern zu testen.

! Rückmeldungen zu Folge, funktioniert die Routine ausschliesslich in **Word2000**.

Links zu diesem Thema

Eigene Links

! Einige der nachfolgenden Links rufen einen ziemlich grossen Download auf.

► Das ganze Resultat als PDF-Datei (750 kb)
► Das ganze Resultat als Word 97/2000-Dokument (3500 kb)
► Eine Kostprobe des Resultates als Word 97/2000-Dokument (300 kb)
► Eine sehr ähnlich Lösung für Excel (HTML-Skript 7 kb)
► Das ganze Resultat für Excel als Excel97/2000-Mappe (1400 kb)

Anmerkung zu den Links

! Das Öffnen der Word- und Excel-Dateien bzw. deren Download kann mehrere Minuten dauern.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Symboll.htm

René Probst, Februar 2000

Änderungen
- --- nil ---

Wie kann ich den Office Assistenten programmieren?

◄ Zurück zum Hauptindex

Ausgangslage

> Ich möchte den Office Assistenten innerhalb eines VBA-Programms dazu benutzen, um den Benutzer bei seiner Arbeit zu unterstützen. Was für Möglichkeit bieten sich mir in diesem Bereich?

Diese Seite

- Ausgangslage
- Ausstattungsmerkmale des Office-Assistenten
- Text oder formatierten Text anzeigen
- Dem Benutzer eine Auswahl anbieten und darauf reagieren
- Den Office-Assistenten animieren
- Auswahlmöglichkeiten auf mehr als fünf Punkte erweitern
- Office Assistent im ungebundenen (nicht modalen) Modus agieren lassen
- Eigene Bitmap oder Metadatei (Grafik) in der Sprechblase anzeigen
- Feststellen, ob der Office-Assistent beim Benutzer (korrekt) installiert ist
- Die wichtigsten Parameters (Eigenschaften) um den Office Assistenten zu steuern
- Links zu diesem Thema

Antwort

Die einen mögen ihn, die anderen hassen ihn. Der Office Assistent, welche die Meinungen derart polarisiert, kann aber sehr wohl dazu benutzt werden, um unsere Programme benutzerfreundlicher zu gestalten. Insbesondere erlaubt er uns, Makros zu schreiben und zu vertreiben, welche trotz integrierter Benutzeroberfläche, das Zeichnen und den Vertrieb von *UserForms* überflüssig machen. UserForms lassen sich bekanntlich im Gegensatz zu VBA-Code nicht direkt vertreiben und verschicken. Entweder muss man Sie mit einer Word-Vorlage weiterreichen oder man exportiert die *UserForm*-Definitionen in eine Datei mit der Erweiterung ***.frm** und weisst den Empfänger an, diese Datei wieder zu importieren. Diese Problematik kann elegant umgangen werden, wenn man statt einer UserForm, wo möglich, den Office Assistenten einsetzt.

Der Office Assistent kann einfachen Text anzeigen...

Ausstattungsmerkmale des Office-Assistenten

▲ Zurück zum Anfang

...formatierter Text in Form von Aufzählungen anzeigen...

Der Office-Assistent kann:

- **Text ausgeben**, ähnlich wie eine *MsgBox* nur ist der opti sche Effekt feudaler.
- **Aufzählungen** ausgeben.
- **Nummerierungen** ausgeben.
- **Auswahl** präsentieren und darauf reagieren.
- Dem Benutzer **Schaltflächen** also *Command Buttons* anzei gen und darauf reagieren.
- **Kontrollkästchen** (*Checkboxes*) ausgeben und darauf rea gieren.
- Kapriolen aufführen, sprich dem Benutzer **Animationen** vor spielen.
- Im Gegensatz zur *UserForms*, welche in **Word 97** immer nur modal (gebunden) sein können, kann der Office Assistent auch im **ungebundenen Modus agieren**, dass heisst er kann vorübergehend den Fokus verlieren, damit sich der Benutzer im gleichen Fenster anderem widmen kann, wie wir dies z.B. vom word-integrierten Dialog "Suchen und Erset zen" kennen.

Text oder formatierten Text anzeigen

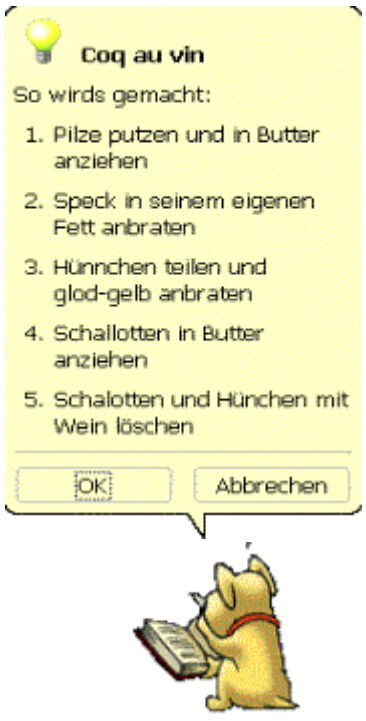
▲ Zurück zum Anfang

Folgender Code ruft den Office Assistenten auf. In einer Art **Sprechblase** wird dem Benutzer, also Ihnen, formatierter Text angezeigt.

Kopieren Sie den ganzen Code, also alle Prozesse, im VBA-Editor in ein **neues** Modul und rufen Sie den Prozess mit dem Namen **AssistentFormatierterTextAnzeigen** auf.

► So bringen Sie das Makro ein und verknüpfen es mit einem neuen Schaltknopf auf der Symbolleiste Ihrer Wahl

```
Private asTitle As String
Private asVisible As Boolean
Private oBlase As Balloon

Sub AssistentFormatierterTextAnzeigen()
  asVisible = Assistant.Visible
  asTitle = "Hallo Welt!"
  Assistant.Visible = True
  Assistant.Animation = msoAnimationAppear
  BlaseAnzeigen
  HappyEnd
End Sub
```



...und formatierter Text in Form von Nummerierungen anzeigen.

```
Private Sub HappyEnd()
  Assistant.Visible = asVisible
  End
End Sub

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeBullets
    .Icon = msoIconTip
    .Heading = asTitle
    asText = "Hallo, ich bin der Office Assistent. & vbCrLf & vbCrLf
    asText = asText & "Ich kann neben Text auch Aufzählungen, Nummerierungen "
    asText = asText & "und Schaltflächen anzeigen." & vbCrLf & vbCrLf
    .Text = asText & "Die Variablen dazu heissen:"
    .Labels(1).Text = "msoBalloonTypeBullets"
    .Labels(2).Text = "msoBalloonTypeNumbers"
    .Labels(3).Text = "msoBalloonTypeButtons"
    .Checkboxes(1).Text = "Ich kann auch Kontrollkästchen anzeigen."
    .Button = msoButtonSetOK
    BlaseAnzeigen = .Show
  End With
End Function
```

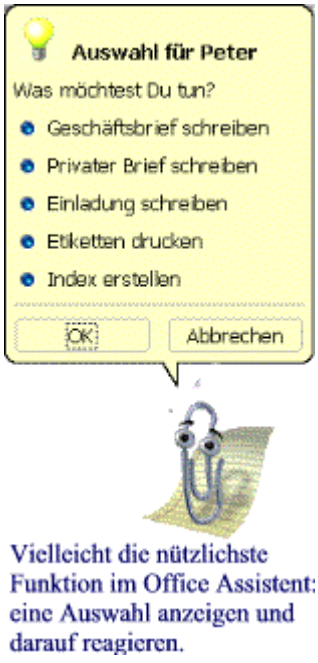
(Der Office Assistent wird hier genutzt, um dem Benutzer in einer Sprechblase formatierten Text anzuzeigen.)

Dem Benutzer eine Auswahl anbieten und darauf reagieren

▲ Zurück zum Anfang

Folgender Code ruft den Office Assistenten auf. In einer Art **Sprechblase** wird dem Benutzer, eine Auswahl von fünf Möglichkeiten angeboten. Zusätzlich bieten zwei Kontrollkästchen Unteroptionen an. Hat der Benutzer gewählt, wird die entsprechende Aktion durchgeführt. Selbstverständlich lassen sich auch die Kontrollkästchen abfragen, um darauf zu reagieren.

Kopieren Sie den ganzen Code, also alle Prozesse, im VBA-Editor in ein **neues** Modul und rufen Sie den Prozess mit dem Namen **AssistentAuswahlAnzeigenUndWahlAusführen** auf.



Vielleicht die nützlichste Funktion im Office Assistent: eine Auswahl anzeigen und darauf reagieren.

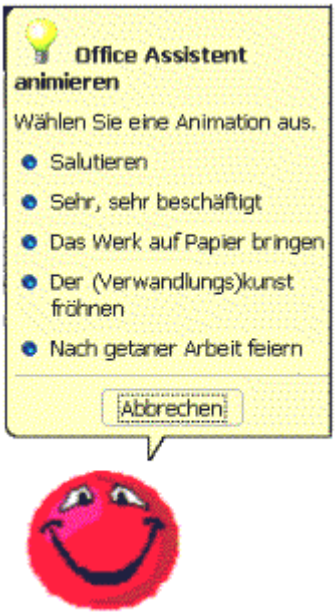
```
Private asTitle As String
Private asVisible As Boolean
Private oBlase As Balloon

Sub AssistentAuswahlAnzeigenUndWahlAusführen()
  asVisible = Assistant.Visible
  asTitle = "Mach es farbig!"
  Assistant.Visible = True
  Assistant.Animation = msoAnimationAppear
  Select Case BlaseAnzeigen
    Case -2 'Der Benutzer hat den Dialog abgebrochen
      HappyEnd
```

```
    Case 1
      DokumentFormatieren wdRed
    Case 2
      DokumentFormatieren wdGreen
    Case 3
      DokumentFormatieren wdBlue
    Case 4
      DokumentFormatieren wdPink
    Case 5
      DokumentFormatieren wdBlack
  End Select
  HappyEnd
End Sub

Private Sub DokumentFormatieren(Farbcode)
  With ActiveDocument.Content.Font
    .ColorIndex = Farbcode
    .Bold = oBlase.Checkboxes(1).Checked
    .Italic = oBlase.Checkboxes(2).Checked
  End With
End Sub

Private Sub HappyEnd()
  Assistant.Visible = asVisible
  End
End Sub

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = asTitle
    asText = "Wählen Sie Schriftfarbe und Schriftstil in der das Dokument "
    .Text = asText & " formatiert werden soll."
    .Labels(1).Text = "Rot"
    .Labels(2).Text = "Grün"
    .Labels(3).Text = "Blau"
    .Labels(4).Text = "Pink"
    .Labels(5).Text = "Schwarz"
    .Checkboxes(1).Text = "Fett"
    .Checkboxes(2).Text = "Kursiv"
    .Button = msoButtonSetCancel
    BlaseAnzeigen = .Show
  End With
End Function
```

(Der Office Assistent bietet hier dem Benutzer fünf Auswahlmöglichkeiten an. Dank zwei Kontrollkästchen, vervielfachen sich die Optionen.)

Den Office-Assistenten animieren

▲ Zurück zum Anfang



Folgendes Beispiel demonstriert, wie man den Office Assistenten aus einem Programm heraus zu Kapriolen verführen kann.

Kopieren Sie den ganzen Code, also alle Prozesse, im VBA-Editor in ein **neues** Modul und rufen Sie den Prozess mit dem Namen **AssistentAnimieren** auf.

```
Private asTitle As String
Private oBlase As Balloon

Sub AssistentAnimieren()
  asTitle = "Office Assistent animieren"
  Assistant.Visible = True
  Select Case BlaseAnzeigen
    Case -2 'Der Benutzer hat den Dialog abgebrochen
      End
    Case 1
      Assistant.Animation = msoAnimationGreeting
    Case 2
      Assistant.Animation = msoAnimationWorkingAtSomething
    Case 3
      Assistant.Animation = msoAnimationPrinting
    Case 4
      Assistant.Animation = msoAnimationGetArtsy
    Case 5
      Assistant.Animation = msoAnimationCharacterSuccessMajor
  End Select
  AssistentAnimieren
End Sub

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = asTitle
    .Button = msoButtonSetCancel
    .Text = "Wählen Sie eine Animation aus."
    .Labels(1).Text = "Salutieren"
    .Labels(2).Text = "Sehr, sehr beschäftigt"
    .Labels(3).Text = "Das Werk auf Papier bringen"
    .Labels(4).Text = "Der (Verwandlungs)kunst fröhnen"
    .Labels(5).Text = "Nach getaner Arbeit feiern"
    BlaseAnzeigen = .Show
  End With
End Function
```

(Der Office Assistent als Ihre Marionette.)

Auswahlmöglichkeiten auf mehr als fünf Punkte erweitern

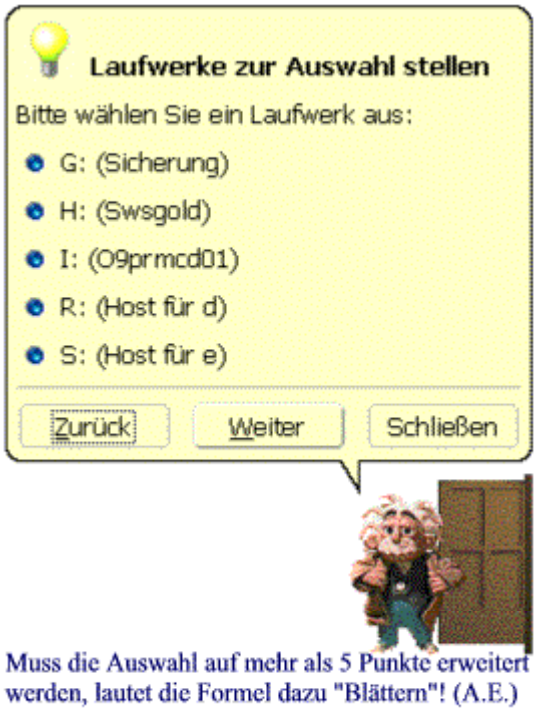
▲ Zurück zum Anfang

Auf den ersten Blick macht es den Anschein, als könnte der Office Assistent nur fünf Wahlmöglichkeiten anbieten, da Sie im Programm ja nur *Label(1)* bis *Label(5)* setzen können. Es ist richtig, dass der Office-Assistent nur fünf Punkt auf einmal anzeigen kann. Allerdings können Sie den Assistenten dahingehen programmieren, dass der Benutzer blättern kann.

Nachfolgendes Beispiel listet alle logischen Laufwerke auf, welches das System kennt (inklusive Diskettenlaufwerk und CD-ROM-Laufwerk). Vorausgesetzt Ihr Rechner verfügt über mehr als fünf solche Laufwerke, können Sie testen, wie es blättert.

Der Code hat einen kleinen Schönheitsfehler. Das CD-ROM-Laufwerk wird nur erfasst, falls es geladen ist. Dies hat aber selbstverständlich nichts mit dem Office Assistenten zu tun.

Kopieren Sie den gesamten Code in ein neues Modul und rufen Sie den Prozess mit dem Namen **AsistentGrosseAuswahl** auf.



Muss die Auswahl auf mehr als 5 Punkte erweitert werden, lautet die Formel dazu "Blättern"! (A.E.)

```
Private Const asTitle = "Laufwerke zur Auswahl
stellen"
Private asVisible As Boolean
Private LW()
Private LWBez()
Private MaxLW As Integer
Private oBlase As Balloon
Private blPoint As Integer
Private blLabel() As String
Private blButton As Integer

Sub AssistentGrosseAuswahl()
  LaufwerkeAbfragen
  Assistant.Visible = True
  Assistant.Animation = msoAnimationBeginSpeaking
  blPoint = 1
  AssignLabels
  Wahl = BlaseAnzeigen
  If Wahl <> "0" Then
    MsgBox "Der Benutzer hat folgendes Laufwerk gewählt:" & vbCrLf & LWBez(Wahl)
  Else
    End
  End If
End Sub

Private Sub LaufwerkeAbfragen()
  j = 0
  For i = 65 To 90
    On Error Resume Next
    ChDrive Chr(i)
    Fehler = Err.Number
```

```
    On Error GoTo 0
    If Fehler = 0 Or i = 65 Then
      j = j + 1
      ReDim Preserve LW(j)
      ReDim Preserve LWBez(j)
      LW(j) = Chr(i) & ":"
      If Fehler = 0 Then
        lab = Dir(Chr(i) & ":\", vbVolume)
        If lab = "" Then
          LWBez(j) = Chr(i) & ":"
        Else
          LWBez(j) = Chr(i) & ": (" + UCase(Left(lab, 1)) & _
             LCase(Right(lab, Len(lab) - 1)) & ")"
        End If
      End If
      If i = 65 And Fehler > 0 Then LWBez(j) = Chr(i) & ": *nicht bereit*"
    End If
  Next i
  j = j + 1
  ReDim Preserve LW(j)
  ReDim Preserve LWBez(j)
  LW(j) = I2
  LWBez(j) = I3
  MaxLW = j
End Sub

Private Sub AssignLabels()
  If MaxLW < 6 Then
    blButton = msoButtonSetCancel
  Else
    Select Case blPoint
      Case 1
        blButton = msoButtonSetNextClose
      Case Else
        If MaxLW < blPoint + 5 Then
          blButton = msoButtonSetBackClose
        Else
          blButton = msoButtonSetBackNextClose
        End If
    End Select
  End If
  ReDim blLabel(6)
  j = 0
  For i = blPoint To blPoint + 4
    j = j + 1
    If UBound(LWBez) >= i Then blLabel(j) = LWBez(i)
  Next i
End Sub

Private Function BlaseAnzeigen()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = msoModeModal
```

```
    .BalloonType = msoBalloonTypeButtons
    .Icon = msoIconTip
    .Heading = asTitle
    .Text = "Bitte wählen Sie ein Laufwerk aus:"
    Flag = False
    While Flag = False
      i = 1
      While blLabel(i) <> ""
        .Labels(i).Text = blLabel(i)
        i = i + 1
      Wend
      .Labels.Count = i - 1
      .Button = blButton
      Wahl = .Show
      If Assistant.BalloonError = True Then Flag = True
      Select Case Wahl
        Case 1 To 5 'Wahl getroffen
          Flag = True
          BlaseAnzeigen = Wahl + blPoint - 1
        Case -5 'Rückwärts
          blPoint = blPoint - 5
          AssignLabels
        Case -6 'Vorwärts
          blPoint = blPoint + 5
          AssignLabels
        Case -2, -12 'Abbrechen oder Schliessen
          BlaseAnzeigen = 0
          Flag = True
      End Select
    Wend
  End With
End Function
```

(Soll der Office Assistent mehr als fünf Punkte zur Auswahl stellen, ist Blättern angesagt. Diese Routine erhält die zum Scrollen benötige Mimik.)

Office Assistent im ungebundenen (nicht modalen) Modus agieren lassen

▲ Zurück zum Anfang

Ein Glanzpunkt des Office Assistenten ist es, dass er auch im ungebundenen, sprich nicht modalen Modus operieren kann. Dies ist um so wichtiger, da **User-Forms in Office 97 nur modal** sein können. Die Art und Weise, wie z.B. der word-integrierte Dialog **Suchen/Ersetzen** sowohl in **Office 97** wie auch in **Office2000** reagiert, können Sie mittels VBA zwar in **Office2000** nicht aber in **Office 97** nachbilden.

Was macht den Suchen/Ersetzen-Dialog so speziell? Nun das Formular kann den Fokus verlieren, der Benutzer kann den Dialog ohne ihn wegzuklicken verlassen, um sich anderen Sachen zu widmen. Er kann sich in der Anwendung frei bewegen und später wieder den Dialog in den Brennpunkt stellen.

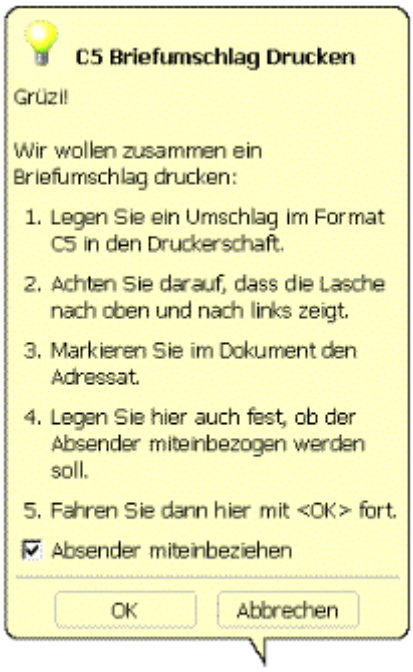
Da der Office Assistent auch in der 97er-Version den nicht modalen Modus offiziell unterstützt, kann er uns dieses Manko ausgleichen.

Das nachfolgende Makro dient dazu, den Benutzer beim Drucken eines Briefumschlages zu unterstützen. Ruft der Benutzer das Makro auf, wird dieser zuerst instruiert, wie der Briefumschlag in das Druckerfach zu legen ist. Zusätzlich wird der Benutzer aufgefordert, den Adressaten in das Dokument zu schreiben bzw. falls dies schon geschehen ist, diesen zu markieren. Der Code endet dann damit der Benutzer sich nun frei in der Word-Anwendung bewegen kann. Allerdings bleibt die Sprechblase des Office Assistenten mit den Instruktionen immer noch sichtbar und der Benutzer kann nach getaner Arbeit jederzeit auf ihn zurückgreifen. Der Benutzer markiert also jetzt den Adressaten, kehrt zur Sprechblase zurück und betätigt dort die Schaltfläche ***Ok***. Der Assistent prüft nochmals ob nun eine Markierung vorhanden ist und druckt dann den Briefumschlag aus. Der Benutzer kann auf diesem Dialog auch angeben, ob die Absenderadresse auch auf dem Briefumschlag erscheinen soll.

Ganz zum Schluss erfolgt dann noch eine Endmeldung, ebenfalls dargestellt mit dem Assistenten, diesmal aber im modalen Modus.

Programmtechnisch wird dies so erreicht, dass der Office Assistent im **nicht modalen Modus** *(modeless)* aufgerufen wird. Zusätzlich wird eine sogenannte **Callback-Prozedur** gesetzt, damit der Assistent weiss, welche Prozedur aufzurufen ist, wenn de Benutzer später dann die Schaltfläche ***Ok*** oder ***Abbrechen*** betätigt. Optional können Sie noch mit der Assistenten-Eigenschaft **Private** arbeiten. Diese Eigenschaft kommt im "Callback-Fall" 1:1 als Variable verpackt, zurück. Dank dieser Variablen können Sie den Kontext bestimmen, falls Sie in einer Abfolge mehrere nicht modale Dialoge ausgeben.

Kopieren Sie den **gesamten** nachfolgenden Code in ein neues Modul und testen und rufen Sie den Prozess mit dem Namen **BriefumschlagDrucken** auf. Testen Sie dann den ungebundenen Dialog der dabei entsteht.



```
' Bedruckt ein C5 Briefumschlag mit Adressat
' und Absender
'
' Erstellt am 02.11.97 von René Probst
' Beispiel für den Einsatz eines nicht modalen Dialoges

Private Const asTitle = "C5 Briefumschlag Drucken" '
Private asVisible As Boolean
Private oBlase As Balloon
Private blType As Integer
Private blMode As Integer
Private blIcon As Integer
Private blTitle As String
Private blText As String
Private blLabel() As String
Private blCheckbox()
Private blButton As Integer
Private blCallback As String
Private blPrivate As Long
Private Wahl As Long
Public BriefumschlagDruckenReturnAddressOmit As Boolean

Sub BriefumschlagDrucken()
  Set oAssistant = Assistant
  If oAssistant Is Nothing Then
    MsgBox "Der Office-Assistent ist nicht oder nicht richtig installiert.", _
       vbCritical, asTitel
  End If
  ReDim blCheckbox(5, 1)
  asVisible = Assistant.Visible
  BriefumschlagDruckenHeisseWillkommen
End Sub

Private Sub BriefumschlagDruckenHeisseWillkommen()
  Assistant.Visible = True
  Assistant.Animation = msoAnimationGreeting
  ReDim blLabel(5)
  ReDim blCheckbox(5, 1)
  blMode = msoModeModeless
  blCallback = "BriefumschlagDruckenCallback"
  blType = msoBalloonTypeNumbers
  blIcon = msoIconTip
  blTitle = asTitle
  blText = "Grüzi!" & vbCrLf & vbCrLf & _
     "Wir wollen zusammen ein Briefumschlag drucken:"
  blLabel(1) = "Legen Sie ein Umschlag im Format C5 in den Drucker."
  blLabel(2) = "Achten Sie darauf, dass die Lasche nach oben und nach links zeigt."
  blLabel(3) = "Markieren Sie im Dokument den Adressat."
  blLabel(4) = "Legen Sie auch fest, ob der Absender miteinbezogen werden soll."
  blLabel(5) = "Fahren Sie dann hier mit OK fort."
  blCheckbox(1, 0) = "Absender miteinbeziehen"
  blCheckbox(1, 1) = True
```

```
  blButton = msoButtonSetOkCancel
  BlaseShow
  End
End Sub

Private Sub BriefumschlagDruckenAdressatPrüfen()
  If Selection.Type <> wdSelectionNormal Then
    BriefumschlagDruckenKeinAdressat
    End
  End If
End Sub

Private Sub BriefumschlagDruckenBriefumschlagDrucken()
  Assistant.Animation = msoAnimationPrinting
  ActiveDocument.Envelope.PrintOut _
  ExtractAddress:=False, _
  OmitReturnAddress:=BriefumschlagDruckenReturnAddressOmit, _
  Height:=CentimetersToPoints(16.2), _
  Width:=CentimetersToPoints(23#), _
  Address:=Selection.Text, _
  AddressFromLeft:=CentimetersToPoints(15), _
  AddressFromTop:=CentimetersToPoints(4), _
  ReturnAddress:="Absenderadresse", _
  ReturnAutoText:="Absenderadresse", _
  ReturnAddressFromLeft:=CentimetersToPoints(1.2), _
  ReturnAddressFromTop:=CentimetersToPoints(14)
End Sub

Private Sub BriefumschlagDruckenKeinAdressat()
  Assistant.Animation = msoAnimationAttentionMajor
  ReDim blLabel(5)
  blMode = msoModeModeless
  blCallback = "BriefumschlagDruckenCallback"
  blType = msoBalloonTypeNumbers
  blIcon = msoIconAlert
  blTitle = asTitle + Chr(13) + Chr(13) + "Kein Adressat markiert!"
  blText = "Leider ist kein Text markiert. - So wirds gemacht:"
  blLabel(1) = "Geben Sie den gewünschten Adressat ein."
  blLabel(2) = "Markieren Sie diesen Text."
  blLabel(3) = "Fahren Sie dann hier mit OK weiter."
  blButton = msoButtonSetOkCancel
  BlaseShow
End Sub

Private Sub HappyEnd()
  Assistant.Animation = msoAnimationCharacterSuccessMajor
  blMode = msoModeModal
  blCallback = ""
  blType = msoBalloonTypeNumbers
  blIcon = msoIconTip
  blTitle = asTitle
  blText = "Das wars!" & vbCrLf & "Au Revoir."
  blButton = msoButtonSetOK
```

```
  BlaseShow
  Assistant.Visible = asVisible
  Selection.Collapse
  End
End Sub

Private Function BlaseShow()
  Set oBlase = Assistant.NewBalloon
  With oBlase
    .Mode = blMode
    If blCallback <> "" Then .Callback = blCallback
    .BalloonType = blType
    .Icon = blIcon
    .Heading = blTitle
    .Text = blText
    For i = 1 To 5
      If blLabel(i) <> "" Then .Labels(i).Text = blLabel(i)
      If blCheckbox(i, 0) <> "" Then
        .Checkboxes(i).Text = blCheckbox(i, 0)
        .Checkboxes(i).Checked = blCheckbox(i, 1)
      End If
    Next i
    .Button = blButton
    .Private = 0
    BlaseShow = .Show
    ReDim blLabel(5)
    ReDim blCheckbox(5, 1)
  End With
  If blMode = msoModeModeless Then
    Assistant.Reduced = True
    Assistant.MoveWhenInTheWay = True
  End If
End Function

Sub BriefumschlagDruckenCallback(oBlase As Balloon, Wahl As Long, Priv As Long)
  oBlase.Close
  ReDim blCheckbox(5, 1)
  ReDim blLabel(5)
  Select Case Wahl
    Case -1
      BriefumschlagDruckenAdressatPrüfen
      BriefumschlagDruckenReturnAddressOmit = Not oBlase.Checkboxes(1).Checked
      BriefumschlagDruckenBriefumschlagDrucken
    Case -2
  End Select
  HappyEnd
End Sub
```

(Der Office Assistent agiert hier im ungebundenen Modus. Damit kann der Benutzer interaktiv bei seiner Arbeit unterstützt werden.)

Anmerkung zum letzten Beispiel

- Wenn der Absender auf dem Briefumschlag nicht erscheint, obwohl Sie die entsprechende Option aktiviert haben, dann liegt dies bestimmt daran, dass auf Ihrem Rechner der AutoText-Eintrag mit dem Namen **Absenderadresse** nicht existiert. Ursprünglich wurde dieser Umstand im Code ebenfalls abgefangen und abgehandelt. Da ich aber das Beispiel nicht überladen wollte, habe ich diesen Teil entfernt. Es ist Ihnen natürlich frei gestellt, einen solchen AutoText-Eintrag, welcher Ihre Adresse enthalten sollte, zu erstellen. Damit sollte der volle Funktionsumfang, welcher der Code bietet, zur Verfügung stehen.

Dies war nun also sozusagen unsere Meisterprüfung. Mehr als in diesem Skript besprochen, können Sie wohl vom technischen Standpunkt aus nicht mehr aus dem Assistenten herauskitzeln. Bestimmt reicht aber Ihre Kreativität dazu aus, sinnvoller Anwendungen mit dem Office Assistenten zu schreiben, als dies in den Beispielen hier der Fall ist.

Eigene Bitmap oder Metadatei (Grafik) in der Sprechblase anzeigen

▲ Zurück zum Anfang

Das letzte, was es noch zu erwähnen gibt, ist die Tatsache, dass Sie ausser einem der vorgesehenen Sinnbilder auch eine eigene Grafik übergeben könnten. Bewusst habe ich auf die nähere Beschreibung dieser Optionen verzichtet. Die Gründe dafür lauten wie folgt:

- Die Grafik muss in einer eigenständigen Grafikdatei vorliegen. Beim Migrieren müssen Sie immer daran denken diese Datei mitzunehmen. Der Pfad zu dieser Datei muss im Makro "hard-coded" sein. Welches Schw... denkt z.B. beim Vertreiben oder beim Zügeln an diese Datei?!
- Die Grafik kann nicht skaliert werden. Es ist auch nicht möglich, den Text um die Grafik fliessen zu lassen. Der zu erzielende optische Eindruck ist also bescheiden.
- Lediglich zwei Grafikformate sind zugelassen, namentlich eine **Bitmap** (BMP) oder eine **Metadatei** (WMF).
- Gewisse Derivate von Bitmaps werden nicht unterstützt. So führen z.B. RLE-komprimierte Bitmaps zum Abschmieren von Word.

Trotzdem hier ein Code-Beispiel aus dem **TechNet** von Microsoft. Grundsätzlich funktioniert der Code.

```
Sub SprechblaseMitEigenerGrafik()
  Set bln = Assistant.NewBalloon
  myBmp1 = "{bmp C:\Windows\Kreise.bmp}"
  myBmp2 = "{bmp C:\Windows\Kugeln.bmp}"
  myText1 = "This text is before the picture, "
  myText2 = " and this text is after the picture."
  myText3 = " {{This is text in braces.}"
  Set bln = Assistant.NewBalloon
  With bln
    .Mode = msoModeAutoDown
    .Heading = myBmp1 & "   Displaying a Bitmap."
    .Text = myText1 & myBmp2 & myText2 & myText3
    .Show
  End With
End Sub
```

(Ihre Sprechblase kann auch eine Bitmap oder eine Metadatei anzeigen.)

Feststellen, ob der Office-Assistent beim Benutzer (korrekt) installiert ist

▲ Zurück zum Anfang

Es gibt immer wieder Benutzer, welche sich vom Office-Assistenten ganz grundsätzlich genervt fühlen. Manche dieser Leute schalten den Office-Assistenten deshalb auf mehr oder weniger qualifizierte Weise aus. Wenn Sie die zahlreichen Beispiele in diesem Skript anschauen, können Sie feststellen, dass der Code mitunter mit einer Mimik beginnt, welche überprüft, ob das Assistenten-Objekt angesprochen werden kann. Falls Sie diese Prüfung unterlassen, und der Office-Assistent ist beim Benutzer nicht (richtig) installiert, dann wird die Prozedur mit einer für den Benutzer wenig aussagenden (sprich sehr technisch orientierten) Fehlermeldung abbrechen. Ich erachte es deshalb als ratsam. die Prüfung durchzuführen. Der Benutzer weiss dann wenigstens, wieso die Routine auf seinem Rechner nicht läuft und man gibt ihm die Möglichkeit, den Assistenten nach zu installieren, oder es ganz bleiben zu lassen.

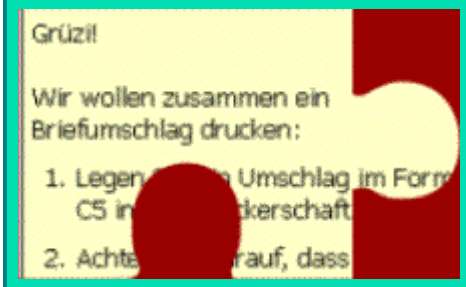
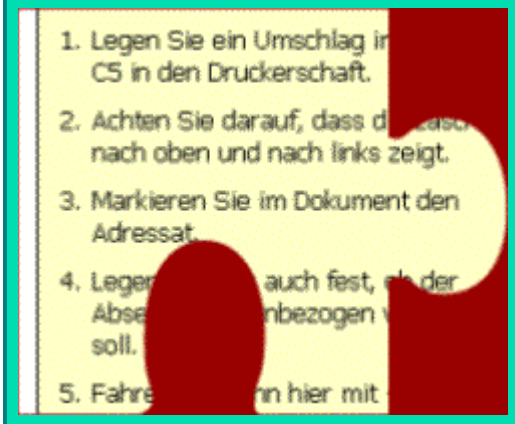
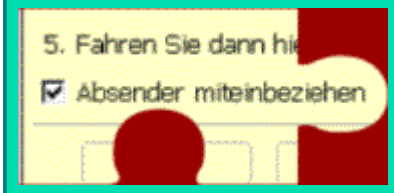
Hier noch einmal die Mimik, welche überprüft, ob das Assistenten-Objekt ansprechbar ist:

```
Set oAssistant = Assistant
If oAssistant Is Nothing Then
  MsgBox "Der Office-Assistent ist nicht oder nicht korrekt installiert.", _
       vbCritical
  End
End If
Set oAssistant = Nothing
```

(Diese Mimik überprüft, ob der Office-Assistent korrekt installiert und damit ansprechbar ist.)

Die wichtigsten Parameters (Eigenschaften) um den Office Assistenten zu steuern

▲ Zurück zum Anfang

Baustein	Bezeichnung	Bedeutung
	Icon	Das Sinnbild oben links in der Sprechblase
	Titel	Der Titel der Sprechblase
	**Text**	Der unformatierte Textbereich der Sprechblase
	**Labels(1).Text bis Labels(5).Text**	Der formatierte Textbereich der Sprechblase in Form einer Aufzählung, einer Nummerierung oder einer Ansammlung von Auswahlmöglichkeiten.
	**Checkboxes(1).Text bis Checkboxes(5).Text**	Kontrollkästchen
	Button	Die Grossen Schaltflächen im Fuss der Blase.

Modus

Eigenschaft	Werte	Bedeutung
Mode	msoModeModal	Der Assistent agiert gebunden
	msoModeModeless	Der Assistent agiert ungebunden

Darstellungsart (Funktion)

Eigenschaft	Werte	Bedeutung
BalloonType	msoBalloonTypeBullets	Der Assistent kann eine Aufzählung anzeigen
	msoBalloonTypeNumbers	Der Assistent kann eine Nummerierung anzeigen
	msoBalloonTypeButtons	Der Assistent kann wählbare Knöpfe (Schaltflächen) anzeigen

Ikone (Sinnbild)

Eigenschaft	Werte	Bedeutung	Bemerkung
Icon	msoIconNone	Der Assistent zeigt kein Ikon an	Der Assistent zeigt kein Ikon an
	msoIconTip	Glühbirne	Glühbirne
	msoIconAlert	Auffälliges Ausrufezeichen	Auffälliges Ausrufezeichen
	msoIconAlertInfo	Ausrufezeichen	Nur **Office2000**
	msoIconAlertWarning	Ausrufezeichen als Verkehrsschild	
	msoIconAlertQuery	Fragezeichen	
	msoIconAlertCritical	Stop Kreuz	

Textelemente und Auswahl

Eigenschaft	Wert	Bedeutung
Title	frei wählbarer Text	Dieser Text erscheint als Titel
Text		Dieser Text erscheint als unformatierter Text im Kopf der Sprechblase
Label(1).Text bis Label(5).Text		Dieser Text erscheint als formatierter Text und zwar je nach Darstellungsart als Aufzählung, Nummerierung oder eine Folge von Befehlsschaltflächen (Auswahl).

Kontrollkästchen

Eigenschaft	Werte	Bedeutung
Checkboxes(1).Text bis Checkboxes(5).Text	frei wählbarer Text	Bezeichnung für Kontrollkästchen
Checkboxes(1).Checked bis Checkboxes(5).Checked	True oder False	Initialisierungswert für Kontrollkästchen

Grosse Schaltflächen setzen

Eigenschaft	Werte	Bedeutung
Button (Schaltflächen setzen)	msoButtonSetNone	Es werden keine Schaltflächen angezeigt Nur für den nicht modalen Gebrauch bestimmt
	msoButtonSetOK	Die Schaltfläche Ok wird angezeigt
	msoButtonSetCancel	Abbrechen
	msoButtonSetOkCancel	Ok und Abbrechen
	msoButtonSetYesNo	Ja und Nein
	msoButtonSetYesNoCancel	Ja, Nein und Abbrechen
	msoButtonSetBackClose	Zurück und Schliessen
	msoButtonSetNextClose	Weiter und Schliessen
	msoButtonSetBackNextClose	Zurück, Weiter und Schliessen
	msoButtonSetRetryCancel	Wiederholen und Abbrechen
	msoButtonSetAbortRetryIgnore	Abbrechen, Wiederholen und Ignorieren
	msoButtonSetSearchClose	Suchen und Schliessen
	msoButtonSetBackNextSnooze	Zurück, Weiter und Erneut erinnern
	msoButtonSetTipsOptionsClose	Tipps, Optionen und Schliessen
	msoButtonSetYesAllNoCancel	Ja, Ja alle, Nein und Abbrechen

Grosse Schaltflächen auslesen

Eigenschaft	Werte		Bedeutung
Button (Wahl des Benuzers)	1		Der Benutzer hat die Auswahl 1 gewählt
	2		Auswahl 2
	3		Auswahl 3
	4		Auswahl 4
	5		Auswahl 5
	-1	msoBalloonButtonOK	Der Benutzer hat die Schaltfläche Ok betätigt
	-2	msoBalloonButtonCancel	Abbrechen
	-3	msoBalloonButtonYes	Ja
	-4	msoBalloonButtonNo	Nein
	-5	msoBalloonButtonBack	Zurück
	-6	msoBalloonButtonNext	Weiter
	-7	msoBalloonButtonRetry	Wiederholen
	-9	msoBalloonButtonIgnore	Ignorieren
	-10	msoBalloonButtonSearch	Suchen
	-11	msoBalloonButtonSnooze	Erneut erinnern
	-12	msoBalloonButtonClose	Schliessen
	-13	msoBalloonButtonTips	Tipps
	-14	msoBalloonButtonOptions	Optionen
	-15	msoBalloonButtonYesToAll	Ja, alle

Links zu diesem Thema

Microsoft

► **MS Office Assistant**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Assi.htm

René Probst, Dezember 1999

Änderungen

- --- nil ---

Wo kann ich den Office2000 Service-Release 1 beziehen?

◄ Zurück zum Hauptindex

Frage

> **Seit ich Office2000 auf meinem Rechner installiert habe, plagen mich einige Fehler im Word2000-Programm. Insbesondere werden Seitenzahlen in der Fusszeile zwar richtig angezeigt aber falsch gedruckt. Im weiteren werden Feldfunktionen beim Öffnen von Word 97 Dokumenten in Word2000 falsch übersetzt. Ich habe gehört, dass der Office2000 Service Release 1 diese Probleme und andere beseitigt. Wo kann ich diesen *Bug-Fix* von Microsoft beziehen?**

Antwort

Knapp ein Jahr nach dem offiziellen Erscheinen von Office2000, ist der *1. Service Release* verfügbar. Geht man davon aus, dass Microsoft eine Wartungspflicht hat, ist dies wirklich eine sehr lange Zeit. Die meisten Änderungen wurden im Februar 2000 sanktioniert, ab April war das Paket dann für die englischsprachige Version verfügbar und Mitte Juni stand dann auch die deutschsprachige Version für den Download bereit.

Zweck eines Service Releases ist es, einerseits aufgetauchte Fehler zu beheben und andererseits Wartungsarbeiten zu tätigen. Eine typische Wartungsfunktion im Bezug auf MS Office ist zum Beispiel die Implementierung eines wirksameren Schutzes gegen Viren, welche sich einer Schwachstelle bedienen, die man ursprünglich nicht erkannt oder unterschätz hat. Service Releases bringen in der Regel keine neuen Features, welche ja wiederum Auslöser von neuen Problemen sein könnten.

Das Update verbessert bekannte Fehler im genannten Produkt, soweit diese von Microsoft als genug gravierend eingestuft wurden und relativ risikolos entfernt werden konnten. Aus gut unterrichteter Quelle weiss ich, dass der innere Kern von *winword*, welcher wohl eine heilige Kuh darstellt, nicht geändert werden durfte. Wer hofft, die häufigen Abstürze würden mit diesem Update verschwinden, wird wohl auch diesmal enttäuscht werden. Die Änderungen betreffen mehrheitlich funktionelle Dinge, welche in der ursprünglichen Fassung nicht oder fehlerhaft arbeiteten.

Erfahrungen von früheren grossen MS Office Service Releases zeigen, dass im erstem Service Release vielleicht ca. 30% der bekannten Fehler korrigiert werden und im zweiten Service Release nochmals etwa 10%. Die restlichen Fehler schlummern gemütlich weiter, stiften Verwirrung und kosten viel wertvolle Zeit. Manchmal hat man sogar den Eindruck, das gewisse Fehler von Microsoft gehegt und gepflegt werden. Das ist leider eine etwas magere Ausbeute, aber immer noch viel besser als gar nichts. Sehr ärgerlich bei der Sache ist, dass es z.B. mit Word2000 möglich sein soll Web-Seiten zu erstellen (was mich kalt lässt, denn es gibt dafür bestimmt geeignetere Produkte), aber dass auch der jüngste Wurf von Microsoft immer noch reproduzierbare Fehler bei für ein Textverarbeitungsprogramm so typischen Aufgaben, wie die Erstellung eines Inhaltsverzeichnisses macht.

Wer bisher sehr zufrieden mit Office2000 war, braucht das Update nicht. Er wird darin keine neuen Features entdecken. (vielleicht mit einer Ausnahme, dass die interaktive Rechtschreibung nun auch in ungeschützten Abschnitten eines teilweise geschützten Dokumentes funktioniert).

Microsoft hat die Änderungen sehr detailliert dokumentiert und zwar in einer sehr übersichtlichen und benutzerfreundlichen Excel-Arbeits-Mappe. Den Link zu dieser englischsprachigen Dokumentation, wie auch die Download-Adresse für den *Patch* finden Sie am Ende dieses Skriptes.

Die Grösse des Bug-Fixes ist so gross wie noch nie. Diese Grösse des Downloads variiert, je nach dem, welches Office-Paket Sie besitzen und welche Office-Funktionen Sie installiert haben. Eine typische Grösse dürfte zwischen **35 - 40 MB** liegen, im Maximum sind es **54 MB**.

Bitte beachten Sie, dass der Service Release auch Teile des Internet Explorers 5 austauscht. Mit dieser grossartigen Innovation von Microsoft, nämlich das Betriebssystem mit dem Web-Browser und den Web-Browser mit Anwendungsprogrammen zu verflechten, müssen wir wohl noch einige Zeit leben. Zumindest so lange, bis das Urteil des US-amerikanischen Justizministerium juristisch und praktisch durchgesetzt werden kann oder bis bei Microsoft die Einsicht einkehrt.

Man rechne typischerweise mit einer Download-Zeit von knapp 3 Stunden und einer anschliessenden Installationszeit von knapp einer Stunde. Der Download läuft in zwei Phasen ab. Zuerst wird die Installationsroutine herunter geladen, welche sehr klein ist. Mit einem Doppelklick auf diese herunter geladene *exe-Datei* und nach gründlicher Überprüfung der Registrierdatenbank durch die Setup-Routine, beginnt dann der grosse Download.

Haben Sie **unbedingt Ihre die Office2000 CD(s) zur Hand**. Download und Installation verlaufen abgesehen von 2-3 einfachen Benutzerinterventionen (CD-ROM einlegen, Neustart des Rechners) vollautomatisch. Es ist übrigens der erste grosse Bug-Fix für MS Office, welchen ich problemlos einspielen konnte. Das **Risiko** das es zu Komplikationen kommt ist allerdings statistisch gesehen ähnlich hoch, wie bei der aller ersten Installation von Office2000, also **relativ hoch**.

Wer den grossen Download scheut, kann das Update auch bei Microsoft bestellen. Das kostet DM 11,00, zuzüglich ein paar Mark für den Versand. Die Lieferungszeit kann ein paar Wochen in Anspruch nehmen. Die Bezugsadresse lautet:

Microsoft Direkt
Postfach 1199
33410 Verl
Tel: 01805/25 11 99
Fax: 01805/25 11 91
E-Mail: microsoft@service.microsoft.de

Hier einige Verbesserungen in MS Word, welche mir ganz persönlich aufgefallen sind. Zuerst diejenigen, welche ich bereits getestet habe:

Verbesserungen

Getestete Verbesserungen

- Das Problem beim Drucken, welches mit vielen Druckern auftrat (Seite 1 von 1, Seite 2 von 2, Seite 3 von 3....) ist behoben.
- Die Rechtschreibprüfung im Hintergrund funktioniert zum ersten Mal in der Geschichte von MS Word in einem ungeschützten Abschnitt eines teilweise geschützten Dokumentes.
- Die SetFocus-Methode bei einer UserForm arbeitet wieder richtig (wie in Word 97).
- Das deutsche Wörterbuch und der Thesaurus wurde aktualisiert. Leider konnte ich bisher keine qualitative Verbesserung bei der Rechtschreibprüfung feststellen.

Nicht getestete Verbesserungen

- Das Datum von nicht in Deutsch geschriebenen Word-Dokumenten wird nun richtig umgesetzt, wenn diese unter der deutschsprachigen Version geöffnet werden.
- Das Semikolon in Formeln (Feldfunktionen) wird in der deutschen Version wieder als Trennzeichen interpretiert.
- JPG-Bilder mit Ränder werden nicht mehr ungewollt in das GIF-Format konvertiert
- Die Schrift Arial Unicode steht jetzt auch bei einer Standardinstallation zur Verfügung.

Bekannte Probleme

Sollten nach dem *Upgrade* auf Service Release 1 alle Ihre Office-Anwendungen nur wenige Sekunden nach dem Start wieder schliessen, dann erkennt die Installationsroutine Ihre Office2000 Lizenz nicht als legitime Lizenz an (z.B. Vorabschau-Lizenz). Der Service Release sollte nur bei voll lizenzierten Produkten installiert werden. Falls Ihnen diese geschieht, deinstallieren Sie Office2000, lesen Sie die *Häufig gestellten Fragen zum Service Release*, welche Microsoft zusammengestellt hat (der Link ist unten eingeblendet) und nehmen Sie allenfalls mit Microsoft direkt Kontakt auf.

Links zu diesem Thema

Microsoft

Englisch

► **SR1Changes.xls (Excel Arbeits-Mappe 320kb)**

Deutsch

► **Häufig gestellte Fragen zu Office2000 Service Release 1a (SR-1a)**

► **Downloaddetails zu Office2000 Service Release 1a (SR-1a) Update**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/O2kSR1a.htm

René Probst, Juni 2000

Änderungen

- --- nil ---

Hilfe! - Mein Word schmiert ab! (Alias: Die dreizehn Gebote, wenn Word spinnt!

◄ Zurück zum Hauptindex

Diese Seite

Frage

> **Wenn ich mein Word-Dokument mit den Befehlen *Ausschneiden* und *Einfügen* bearbeiten will, schliesst sich Word unvermittelt und es wird dann nur noch eine Meldungsbox angezeigt, welche aussagt, dass die Anwendung wegen eines ungültigen Vorgangs geschlossen wurde. Anschliessend muss ich den Rechner neu Aufstarten, um wieder vernünftig arbeiten zu können.**

Vorgehen

Bleiben Sie ruhig, atmen Sie tief durch und arbeiten Sie **systematisch** folgende vier Schritte durch:

▼ Festhalten der Fehlermeldung

▼ Liste von bekannten und verbreiteten Abstürze in MS Word

▼ Schema um unspezifischen Abstürzen von MS Word zu begegnen

▼ Anmerkungen zu Abschmierereien von MS Word

▼ Der Umgang mit defekten Dokumenten

Liste von bekannten und verbreiteten MS Word Abstürze

▲ Zurück zum Anfang

Ich bin mir bewusst, das die folgende Liste noch etwas mager ist. Ich habe Abstürze, mit denen ich mich in der Vergangenheit herumgeschlagen habe, nicht systematisch gesammelt, sondern zitiere diese aus meiner langjährigen Erfahrung mit MS Word.

Auch ich beschäftige mich eben lieber mit konstruktiven Sachen, als mit Abstürzen. Die Liste hat aber schon eine gewisse Relevanz und soll in Zukunft laufend erweitert werden; jetzt wo der Anfang mal gemacht ist.

Sollten Sie in der folgenden Tabelle keine Entsprechung für den Fehler finden, welche auf Ihrem Rechner auftritt, dann fahren Sie mit dem gleich anschliessenden Kapitel fort, welches aufzeigt, wie man einem nicht spezifischen Absturz von MS Word begegnet

Bei welcher Handlung kracht es?	Symptome	Betroffene Programm-komponente (Modul)	Wahrscheinliche Ursache	Lösungsweg	Dokumen-tation von Microsoft	Schuldfrage
Ausschneiden Einfügen	Wenn Sie mit den Tastenkom-binationen [Strg+X], [Strg+C] [Strg+V] arbeiten schmiert Word nicht immer - aber immer öfters - ab.	OSA8.DLL OSA9.DLL WINWORD.EXE	Meistens sind Hintergrundpro-gramme dafür verantwortlich. Auch eine defekte Normal.dot oder der Grafikkartentreiber kommt als Sünder in Frage.	Lesen Sie dazu das Kapitel: ▼ **Probleme mit ausschnei-den, kopieren, einfügen**	▶ **Q224611**	Programme von dritter Seite. - oder - Grafikkarten-treiber schlecht abgestimmt mit MS Word bzw. v.v.
Formel-editieren	Beim Editieren eines **Formel Editor** Objektes schmiert Word reproduzierbar ab.	EQNEDT32.EXE	In den Formatvorla-gedefinitionen des Formel-Editors fehlen die Schrift-artangaben.	Lesen Sie dazu folgendes Skript von Microsoft: ▶ **Ungültige Seite wegen Symboleingab e im Formel-Editor3.0**	▶ **D37785**	Microsoft
Wechseln in die Seitenlay-out-Ansicht oder Seiten-ansicht. Drucken von Dokumenten.	Immer oder oft, wenn Sie in eine der genannten Ansichten wechseln schmiert MS Word ab - oder - Es kracht beim Drucken.	Diverse z.B. HPxxxxx.xxx	Dieser Fehler hängt vermutlich mit dem Druckertreiber zusammen.	Lesen Sie dazu das Kapitel: ▼ **Absturz beim Drucken und druckver-wandten Funktionen**	---	Druckertreiber schlecht abge-stimmt mit MS Word oder v.v.
Datei öffnen	Wenn Sie eine bestimmte Datei öffnen, schmiert Word reprodu-zierbar ab.	Diverse	Die Datei ist wahr-scheinlich defekt.	Lesen Sie dazu das Kapitel: ▼ **Umgang mit defekten Do-kumenten**	▶ **D37785**	Dateikorruption wird verursacht durch instabile Betriebssyste-me, instabile Anwendungs-software, gra-vierende Soft-warefehler oder durch unsach-gemässe Hand-habung des Benutzers
Formularfeld (Eigenschaf-ten/Optionen anzeigen)	Wenn Sie den Formularschutz aufheben und den Eigeschafts-dialog eines Formularfeldes anzeigen las-sen, schmiert Word reprodu-zierbar ab.	MSO97.DLL	VBA-Prozesse mit dem selben Namen kommen in ver-schiedenen Modu-len vor. Mit dieser regulären Situation kommt Word 97 nicht zu recht.	Lesen Sie dazu folgendes Skript von Microsoft: ▶ **Word stürzt ab beim Anzeigen von Formularfeldei gen-schaften**	▶ **D42774**	Microsoft Nach Angaben von Microsoft in Word2000 behoben.
Dokumentva-riable (Referenz setzen)	In Word 97, bei Verwendung des Feldes DocVaribale in der Kopf-/ Fuss-zeile bzw. Text-feld, schmiert Word reprodu-zierbar ab.	WINWORD.EXE	Lausige Program-mierung in MS Word	Weichen Sie auf **Dokumenteigen-schaften** aus, welche fast den identischen Zweck abdecken können..	▶ **Q168923**	Microsoft Das Problem ist in Word2000 gelöst.

Bei welcher Handlung kracht es?	Symptome	Betroffene Programm-komponente (Modul)	Wahrscheinliche Ursache	Lösungsweg	Dokumen-tation von Microsoft	Schuldfrage
VBA6 Hilfe-themen (aufrufen oder drucken)	Beim Aufruf ganz bestimmter VBA-Hilfethemen schmiert Word2000 reproduzierbar ab.	MSOHELP HHCTRL.OCX	Schlechte Pro-grammierung von Microsoft.	Schlagen Sie die betroffenen Hilfe-themen im Internet (TechNet) nach.	▶ **Q244354**	Microsoft
VBA (entwickeln von Code)	Wenn Sie lange und intensiv VBA-Code entwickeln, schmiert Word des öfteren ab.	VBA332.DLL	Wahrscheinlich schlechtes Spei-cher-management von Windows.	Sichern, sichern und nochmals sichern.	---	Microsoft
VBA (Umbenennen einer User-Form)	Nennt man eine UserForm zu einem Namen um, welchen es vorher gab ohne das Dokument zwischenzeitlich zu schliessen, schmiert Word ab.	VBE6.DLL	Oberlausige Pro-grammierung von MS Office2000	Speichern und neu Öffnen, dann Aktion wiederholen	Nicht doku-mentiert	Microsoft
Symbolleis-tenschaltflä-chen (kopieren und einfügen)	Die Handha-bung von eini-gen bestimmten Symbolen mit der CopyFace- bzw. PasteFace-Methode in VBA, bringt Word 97 und Word2000 zum abschmieren.	VBA332.DLL	Unsaubere Formate oder unsaubere Programmierung	Auf diese Symbole verzichten	---	Microsoft
Shapes SourceName-Eigenschaft	Spricht man die SourceName- oder Source-FullName-Eigenschaft einer Shape ex VBA aktiv an, dann geht sofort der Speicher aus und wird auch nicht mehr frei gegeben. Word schmiert dann ab.	Diverse	Lausige Program-mierung Lausiges Spei-chermanagement von Windows.	Keine Umgehungs-möglichkeit	▶ **Q176995**	Microsoft Angeblich soll das Problem in WD97 auftau-chen und in WD98 geflickt sein. In WD2000 jedenfalls be-steht das Prob-lem weiterhin.
UserForm (Elemente platzieren)	Fügen Sie einer UserForm mehr als 411 Elemente zu, schmiert Word reprodu-zierbar ab.	<Unbekannt>	Schlechtes Pro-duktdesign Lausige Fehlerab-handlung.	Es gibt keine Lösung. In Office2000 scheint eine höhere Limite zu gelten.	▶ **Q177842**	Microsoft

Schema um unspezifische Abstürze der Word-Anwendung zu begegnen

▲ Zurück zum Anfang

! Die Reihenfolge der folgenden Aktionen ist nicht zufällig gewählt. Wenn nicht triftige Gründe dagegen oder für ein anderes Vorgehen sprechen, gehen Sie bitte in der vorgeschlagenen Reihenfolge vor. Selbstverständlich kann die Übung abgebrochen werden, sobald die Word-Anwendung wieder richtig muckt.

	Priorität	Was ist zu tun?	Wie wird es erreicht?	Wieso muss ich das tun?	Bemerkungen
1	**sehr hoch**	Stellen Sie die Fehlermeldung im **Originalwortlaut** sicher.	Lesen Sie dazu das Kapitel: ▼ **Wie halte ich die Fehlermeldung fest**	Ohne Fehlermeldung im **Originalwortlaut** kann Ihnen nicht gezielt geholfen werden.	Aus der Fehlerbeschreibung muss mindestens das **Modul**, welche den Fehler ausgelöst hat, hervorgehen. Zusätzlich muss bekannt sein, ob es sich sich bei der Art des Fehlers um eine **«Ungültige Seite»** oder um eine **«Allgemeine Schutzverletzung»** handelt. **Dr. Watson Meldungen sind wertlos!** Lesen Sie dazu das Kapitel: ▼ **Unbrauchbare Fehlermeldungen**
2	**sehr hoch**	Starten Sie Ihren Rechner im abgesicherten Windows-Modus und testen Sie, ob das Problem auch in diesem reduzierten Modus auftaucht.	Lesen Sie dazu die Kapitel: ▼ **Rechner im abgesicherten Windows-Modus starten** ▼ **Feststellen, ob ein Konflikt mit einem Hintergrund Programm besteht**	Es muss abgeklärt werden, ob ein Treiberproblem oder ein Problem mit Hintergrundaktivitäten besteht.	Nicht anwendbar für **Windows NT**. Lesen Sie dazu die Kapitel: ▼ **Windows NT**
3	**sehr hoch**	Nennen Sie die Word-Standardvorlage mit dem Namen «Normal.dot» um.	Lassen Sie nach einer Datei mit dem Namen «Normal.dot» suchen. Nenne Sie diese Datei in «Alt.dot» um. Sollten Sie mehrere Dateien mit diesem Namen finden, nennen Sie alle um.	Die Datei mit dem Namen «Normal.dot» ist für Word vital. Sie wird bei jedem Start von Word geladen und bleibt über die ganze Sitzung geladen. Diese Datei ist hoch gefährdet und geht immer mal wieder kaputt. Dies wiederum kann Word zum Abschmieren bringen.	Word wird diese Datei beim nächsten Start neu anlegen, wenn sie nicht vorhanden ist. Sie können die Bausteine, welche in der Normal.dot enthalten sind, später wieder zurück gewinnen. Lesen Sie dazu folgendes Kapitel: ▼ **Bausteine aus der defekten «Normal.dot» zurück gewinnen**

4	**sehr hoch**	Schliessen Sie Word und nennen Sie folgenden Schlüssel in der Registrierdatenbank nach *Data1* um:
		`HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data`
		Word 97 bzw.
		`HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Data`
		Word2000
			Lesen Sie dazu das Kapitel:
			▼ **Word-Data Schlüssel in der Registrierdatenbank umbenennen**
				Dieser Schlüssel enthält Informationen, welche für Word vital sind. Es handelt sich um eine ellenlange Zeichenfolge im binären Format. Diese Daten werden vom Gespann Windows und MS Word immer mal wieder kaputt geschlagen.	Word wird diesen Schlüssel beim nächsten Start neu initialisieren, wenn er nicht mehr vorhanden ist.
					Sollten Sie sich dafür interessieren, welche Einstellungen dort gespeichert sind, dann lesen Sie das entsprechende Skript, auf welches am Ende dieser Seite verweisen wird.

	Prio-rität	Was ist zu tun?	Wie wird es er-reicht?	Wieso muss ich das tun?	Bemerkungen
5	**sehr hoch**	Testen Sie, ob das Dokument und die Dokumentenvorlage, auf welchem das Dokument basiert, auf einem anderen Rechner ebenfalls einen Absturz verur-sacht.	Lesen Sie dazu das Kapitel: ▼ **Umgang mit defekten Doku-menten**	Selbsterklärend	Falls der Fehler auch auf einem anderen Rechner auftritt, kön-nen Sie die Übung abbrechen. Holen Sie das Dokument von der letzten Datensicherung zurück.
6	**Hoch**	Löschen Sie sämtliche tempora-ren Dateien auf Ihrem Rechner.	Lassen Sie nach Dateien mit der Datei-erweiterung ***.TMP** suchen und löschen Sie möglichst alle davon.	Einige der temporären Dateien, werden bei einem Neustart von Word geöffnet. Oft sind temporäre Dateien aber defekt. Defekte Dateien können Word zum Abschmieren bringen.	Es wird Ihnen wahrscheinlich nicht gelingen, alle temporären Dateien zu löschen, da zwei oder drei davon von Windows allokiert sind. Löschen Sie einfach so viele wie möglich.
7	**Mittel**	Rufen Sie im **Office2000** War-tungsmodus (*Setup*) die Funktion **Office reparieren** auf. **Office 97** Benutzer sollten MS Word deinstallieren und **neu installieren**.	Lesen Sie dazu die Kapitel: ▼ **Office2000 repa-rieren** bzw. ▼ **Office 97 neu installieren**	Microsoft ist sehr stolz auf diese Reparatur-funktion in **Office2000** (wahrscheinlich die Marketingabteilung etwas mehr; die tech-nische Abteilung etwas weniger.) **Office 97** Benutzer können noch nicht auf diese Einrichtung zurückgreifen.	Das Reparieren bzw. die Neu-installation beseitigt weniger Probleme, als man gemein hin denkt. Allerdings gehen auch viel weniger Einstellungen dabei verloren, als oft ange-nommen wird. Ersetzt werden dabei fehlende Komponenten, defekte Kompo-nenten dagegen, werden nur in seltenen Fällen erkannt.

8	**Mittel**	Schliessen Sie Word und nennen Sie folgenden Schlüssel in der Registrierdatenbank nach *Opti-ons1* um:
		`HKEY_CURRENT_USER\ Software\Microsoft\ Office\8.0\ Word\Options`
		Word 97 bzw.
		`HKEY_CURRENT_USER\ Software\Microsoft\ Office\9.0\ Word\Options`
		Word2000
			Dieser Schlüssel enthält weitere vitale Einstellungen für MS Word, aller-dings in transparenter und editierbarer Form.	Lesen Sie dazu das Kapitel:
				▼ **Word-Options Schlüssel in der Registrier-datenbank umbe-nennen**
					Für mein Geschmack hat diese Aktion keine grosse Priorität. Der technische Support von Microsoft in München verlangt dies aber in den meisten Fäl-len.

9	**Hoch**	Lassen Sie nach allen Dateien mit der Dateierweiterung ***.ACL** und ***.DIC** suchen. Nennen Sie diese Dateien um.	Lesen Sie dazu das Kapitel: ▼ **AutoKorrektur-Datein und Be-nutzer-wörterbücher umbenennen**	Defekte AutoKorrektur-Dateien und defekte Benutzerwörterbücher können MS Word zum Abschmieren verleiten.	Dieser Schlüssel wird von Word beim nächsten Start neu erstellt, falls Sie nicht vorhan-den sind.
10	**Hoch**	Stellen Sie auf Grund der Feh-lermeldung, den Namen der Programmkomponente (**Modul**) fest, welches in den Absturz involviert ist, und stellen Sie an Hand dieses Indizes und andere Beobachtungen fest, ob der Fehler auf der *knowledge base* von Microsoft dokumentiert ist. - oder - Veröffentlichen Sie - gut doku-mentiert - den Fehler in einem geeigneten Forum.	Lesen Sie dazu das Kapitel: ▼ **Hilfe von der Microsoft Online-Datenbank** ▼ **Empfehlens-werte Foren und News-Gruppen**	Ungefähr 35-40% der Abstürze können, wenn die Fehlermeldung im Originalwortlaut wie-dergegeben wird, gezielt angegangen werden.	Sie brauchen die Detail-Informationen des Fehlers. Es muss klar hervorgehen, wel-ches **Modul** den Fehler ausge-löst hat und ob es sich um einen IPF (Ungültige Seite) oder um einen GPF (Allgemei-ne Schutzverletzung) handelt. **Dr. Watson-Meldungen unter Windows /NT sind wertlos!**

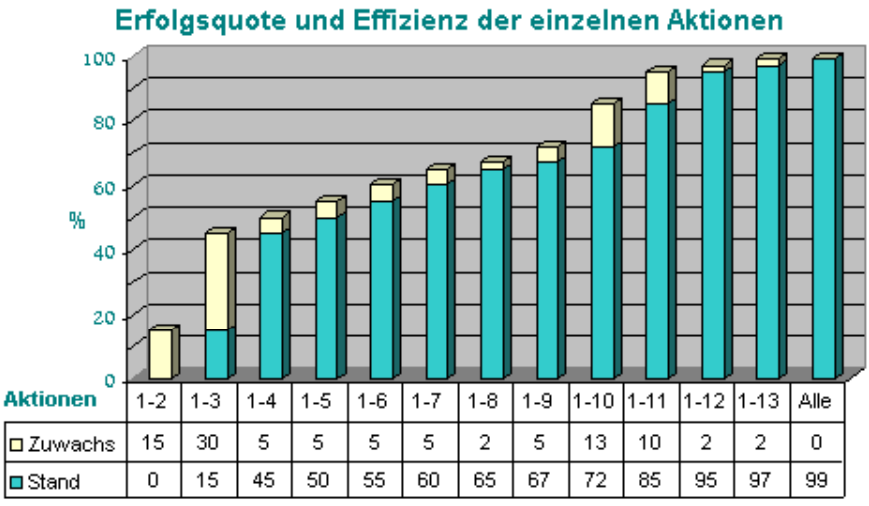
	Prio-rität	Was ist zu tun?	Wie wird es er-reicht?	Wieso muss ich das tun?	Bemerkungen
11	**Hoch**	Lassen Sie Ihre Kollegen und Kolleginnen mal ran. Ziehen Sie einen professionellen Störungsdienst, wie z.B. der von Microsoft in München hinzu.	Lesen Sie dazu das Kapitel: ▼ **PC «Hot Lines»**	Weil alles bisherige nicht zum Erfolg ge-führt hat.	Für Kunden, welche eine MS Office Lizenz direkt von Micro-soft besitzen sind mindestens zwei Anrufe (fast) kostenfrei. Im übrigen rechnen Sie mit einer Preisspanne von DM 35,00 bist DM 185,00 pro Schadenfall.
12	**Mittel**	Entfernen Sie MS Office mög-lichst gründlich von Ihrem Rech-ner und beginnen Sie von vorne.	Microsoft stellt dafür den «**Eraser97**» bzw. den «**Eraser2000**» zur Verfügung. Lesen Sie dazu fol-gendes Skript: ▼ **Office 97/2000 möglichst voll-ständig entfer-nen**	Selbsterklärend.	Wann kommt wohl der Office Terminator?
13	**---**	a) Überprüfen Sie Ihren Rechner mit einem aktuellen **Virenscan-ner**. b) Sichern Sie alle persönlichen Daten und formatieren Sie die Festplattenpartition, auf welcher Windows installiert ist, neu. Installieren Sie dann sämtlich Programme neu.	Diese Aktionen sollten ausschliesslich von fachkundigen Perso-nen ausgeführt wer-den.	Selbsterklärend.	---

Anmerkungen zu Abschmierereien von MS Word

▲ Zurück zum Anfang

Erfolgsquote (Aufwand/Nutzen) bei Befolgung des Ablaufschemas, bzw. Teile davon

Die Praxis zeigt, dass wenn auch nur einzelne der vorgeschlage-nen Aktionen motiviert und sorg-fältig durchgeführt werden, eine hohe Erfolgschance besteht. Die folgende Grafik veranschaulicht nochmals die Effizienz der ein-zelnen Aktionen.



Aktionen	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10	1-11	1-12	1-13	Alle
Zuwachs	15	30	5	5	5	5	2	5	13	10	2	2	0
Stand	0	15	45	50	55	60	65	67	72	85	95	97	99

Es gibt im übrigen auch wenig Alternativen zu diesem systema-tischen und etwas beschwerli-chen Vorgehen. Die Palette der Fehler ist dermassen vielfältig und jede Umgebung präsentiert sich in einem oder anderem De-tail einzigartig.

Die Behauptung von Microsoft und Ihr nahe stehenden Kreisen, dass die zahlreichen Abstürze mehr-heitlich durch fehlerhafte Gerätetreiber hervorgerufen werden, lässt sich durch diese Auswertung jedenfalls kaum erhärten. Das selbe trifft übrigens zu, wenn man die hier publizierte Liste der be-kannten Abstürze studiert.

Vielmehr könnte man als Fazit anführen, dass letztlich sehr viele Komponenten und Ebenen zusam-men spielen müssen, damit die MS Word-Anwendung ohne Abstürze funktioniert. Insbesondere fol-gende Ursachen schlagen in der Praxis fast gleichermassen zu Buche:

Die Verursacher von Abstürzen

- **Drucker- und Grafikkartentreiber**, welche der Anwendungssoftware hinten nachhinken, *bluffen* - also mehr erreichen wollen als verdaubar ist - oder mangelhaft und flüchtig codiert sind.
- Eine defekte bzw. angeschlagene **Standardvorlage** (*Normal.dot*). Dabei stellt sich eine Art Teufelskreis ein. Wer viele Abstürze zu beklagen hat, muss auch damit rechnen, dass diese für Word vitale Datei, oft zerstört wird. Das Konzept von MS Word ist dermassen, dass die *Normal.dot* über die ganze Word-Sitzung geladen sein muss und (dies in der Regel ohne Schreibschutz). Diese Tatsache erhöht die Wahrscheinlichkeit, dass der Datei was zustösst ganz erheblich.
- Korrupte Einstellungen in der **Registrierdatenbank**. Auch hier können wir einen Teufelskreis erkennen. Wenn Word (oft) abschmiert, ist die Integrität dieser Einstellungen nicht mehr gesichert. Wieso Microsoft, solch lange Zeichenfolgen in der Registrierdatenbank erstellt (*Data-Schlüssel*), wobei diese Zeichenfolge sowohl vitale Elemente, wie auch Nebensächlichkeiten enthält (z.B. die Liste der zuletzt geöffneten Dateien), ist ein Rätsel und Ärgernis.
- **Qualitative Defizite** von MS Word. Auf Grund meiner zwei jährigen Erfahrung mit Word-VBA Programmierung und anderen Indizien, bin ich überzeugt, dass es in MS Word nur so wimmelt von Fehlern und Schwachstellen.
- **Defekte Word-Dateien** und Behelfsdateien. Auch defekte Dateien sind oft die Folge von unkontrollierter Beendung der Programme (also Abstürze) von MS Word oder Windows.

Die Hintergründe von Abstürzen

Durchleuchtet man das Problem sowohl etwas tiefer wie auch kausaler und analytischer kommen folgende Hintergründe zum Vorschein:

- Die Tücken einer WYSIWYG-Anwendung (*What You See Is What You Get*) wurde in den 80er-Jahren, als diese Methode aufkam und sehr viele Anhänger fand, unterschätzt. Die fast endlose Flut von Probleme hat man auch heute nicht im Griff.
- Das enorm komplexe Dateiformat, bedingt durch das *WYSIWYG*-Konzept und der äusserst reichhaltigen Funktionalität, welche der Benutzer erwartet.
- Zahlreiche Zusätze, welche wenig mit Textverarbeitung zu tun haben, die MS Word aber zu einem Massenprodukt machen (im doppelten Sinne des Wortes).
- Schnellelebigkeit von Anwendungssoftware und Gerätetreibern.
- Komplexität der Textverarbeitung bzw. ihre Umsetzung mittels elektronischer Verarbeitung.
- Starke Defizite was Stabilität und Qualität von MS Word betrifft.
- Mangelnde Stabilität der Betriebssysteme.
- Vernachlässigung der Softwarewartungspflicht seitens Microsoft.
- Fehlende «*Coverage*» in den Betrieben für Produkte, wie MS Word. Unter «Coverage» verstehe ich, dass es eine Gruppe von motivierten und kompetenten Personen geben müsste, deren Aufgabe es nicht ist, schöne Sachen mit MS Word auf die Beine zu stellen, sondern deren Job es ist, Schwachstellen rund um diese Anwendung *pro-aktiv* zu erkennen und zu regeln, bevor diese eskalieren.
- Oftmals fehlende, professionelle Datensicherung, wie man dies im Grossrechnerumfeld kennt.
- Mangelndes Verständnis vieler Benutzer, wie fragil eine komplex formatierte Datei ist.
- Mangelnde Einsicht beim Benutzer, dass eine eierlegende *Wollmilchsau* selten ein wirklich gutes Profi-Werkzeug darstellt.
- *Last but not least* das de facto Monopol von Microsoft in diesem Bereich.

Rückschlüsse aus dem Namen des Moduls auf die fehlerhafte Komponente ziehen

▲ Zurück zum Anfang

Das Festhalten des Moduls, welches in den Absturz involviert ist, ist auch deshalb wichtig, weil in vielen Fällen Rückschlüsse gemacht werden können, welche Komponente der Anwendung betroffen ist. Oft drücken z.B. die ersten zwei Buchstaben im Namen des Moduls aus, zu welchem Teil der Anwendung dieses Modul gehört. Die nachfolgende Tabelle zeigt die soeben beschriebene Verbindung auf.

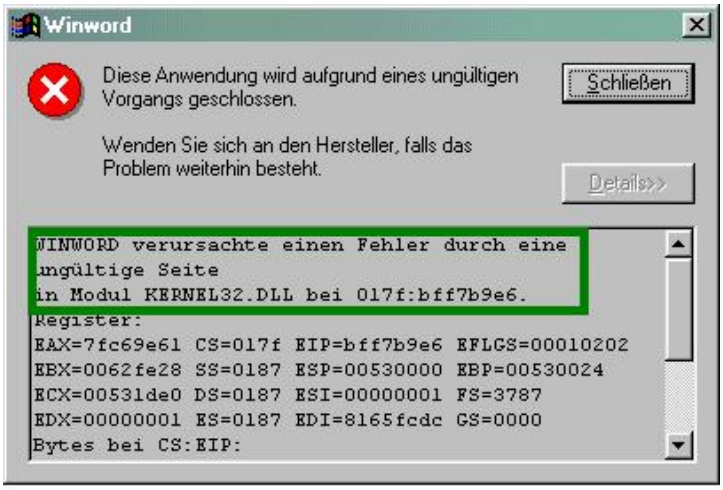
Präfix des Moduls	Komponente	Bemerkungen
KERNEL32.DLL	Windows	Suchen Sie das Problem eher in Umfeld Betriebssystem statt in MS Office oder MS Word. Diese Programmbibliothek ist für Windows äusserst vital. Am Ende dieses Skripts finden Sie einen Link zu einer Abhandlung von Microsoft, welche sich speziell mit den zahlreichen Fehlern befasst, welche im Zusammenhang mit dieser Programmbibliothek stehen..
USER.EXE	Windows genauer den Umgang mit Fenster unter Windows	Das Fehlverhalten steht wohl eher im Zusammenhang mit Windows als mit Word. Versuchen Sie Windows neu (darüber) zu installieren.
GDI.EXE	Grafikkartentreiber	GDI steht für Graphical Display Interface. Dieses Skript hält zahlreiche Tipps bereit, was beim Verdacht eines Problems mit dem Grafikkartentreiber zu tun ist. Wenn Sie nicht Windows NT einsetzen, dann starten Sie den Rechner im angesicherten Windows-Modus um zu testen, ob der Fehler auch dann - unter Ausschluss von komplexen Treibern auftritt. Sie sollten sich auch nach einer neueren Version des Treibers umschauen. Falls bereits der neuste Grafikkartentreiber verwendet wird, installieren Sie diesen nochmals.
OSAxxxxx	MS Office	Sie haben zusätzlich Pech. Der Name des Moduls lässt kaum Rückschlüsse auf den Fehler zu. Suchen Sie das Problem eher im Umfeld von MS Office, als in MS Word selbst
WINWORD.EXE	MS Word	Sie haben zusätzlich Pech. Bei der winword.exe handelt es sich um die mit Abstand wichtigste Programmkomponente von MS Word. Detaillierte Rückschlüsse auf die Komponente, welche den Fehler verursacht, sind deshalb auf Grund des Modulnamens nicht möglich.
VBAxxxxx **VBExxxxx**	VBA Umgebung	Das Problem ist im Zusammenhang mit der Entwicklung oder mit dem Gebrauch von Makroelementen zu sehen. Treiben Sie den Autor, welcher solche Komponenten bei Ihnen implementiert hat, auf.
FMxxxxxx	MS Forms	Das Problem ist im Zusammenhang mit der Entwicklung oder mit dem Gebrauch von User-Forms zu sehen. Treiben Sie den Autor, welcher solche Makroelemente bei Ihnen implementiert hat, auf.
CGxxxxx	ClipArt	Das Problem steht im Zusammenhang mit Microsoft ClipArt Gallery.
EQxxxxxx	Formel Editor (Equation Editor)	Das Problem steht im Zusammenhang mit dem Microsoft Formel Editor.
Orgxxxxx	Organigramm	Das Problem steht im Zusammenhang mit Microsoft Organigramm.
MSSPxxxx	Rechtschreibprüfung	Das Problem steht im Zusammenhang mit der Rechtschreibprüfung.
MSHYxxxx	Silbentrennung	Das Problem steht im Zusammenhang mit der Silbentrennung.
MSGRxxxx	Grammatikprüfung	Das Problem steht im Zusammenhang mit der Grammatikprüfung..
HPxxxxxx	Druckertreiber	Sie setzen einen HP-Drucker ein, oder haben früher ein solches Gerät eingesetzt. Sie sollten die Druckersoftware (Treiber) deinstallieren und falls der HP-Drucker immer noch eingesetzt wird neu installieren. Noch besser, beschaffen Sie sich den neusten Treiber von der HP Home Page und installieren Sie diesen.
<unbekannt>	???	Ja, auch das gibt es bei Microsoft. Sie haben Pech, die Fehlermeldung hilft Ihnen in diesem Falle kaum dabei, das Problem einzukreisen.

Wie halte ich die Fehlermeldung fest

▲ Zurück zum Schema
▲ Zurück zum Anfang

Sie können die gesamten Fehlermeldung, also mit den oft wichtigen Detailinformationen, wie folgt festhalten:

1. Bleiben Sie ruhig - atmen Sie dreimal tief durch!
2. Klicken Sie auf der Meldungsbox den Schaltknopf mit der Bezeichnung **Details**.
3. Markieren Sie den gesamten Text, welcher nun zusätzlich in der Meldungsbox erschienen ist, mit der Maus.
4. Mit der Tastenkombination **[Strg + C]** kopieren Sie den markierten Text in die Zwischenablage.



Nach dem Klick auf die Schaltfläche «DETAILS» erscheint dann die relevante Information.

Wichtig für die Fehlerbehebung ist der grün eingerahmte Teil.

5. Fügen Sie nun den Inhalt der Zwischenablage in eine Textdatei oder in ein neues Word-Dokument ein. Speichern Sie dann diese Datei und bewahren Sie die Datei gut auf, bis der Fehler endgültig bereinigt ist.

Anmerkungen

! Posten Sie **keine «*screen shots*»** in einem Forum oder in einer News-Gruppe. Vor allem; fügen Sie keine Bitmaps (BMP) und dann noch gleich in einer 24-Bit Auflösung an. Es belastet das Datenvolumen sehr, entspricht dieses doch ca. dem hundertfachem!!! eines konventionellen Postings. Der Informationswert einer solchen Grafik, ist nicht grösser, als wenn Sie den Inhalt der Fehlermeldung als Text hinein kopieren.

! Lesen Sie bitte auch das folgende Kapitel *Unbrauchbare Fehlermeldungen*

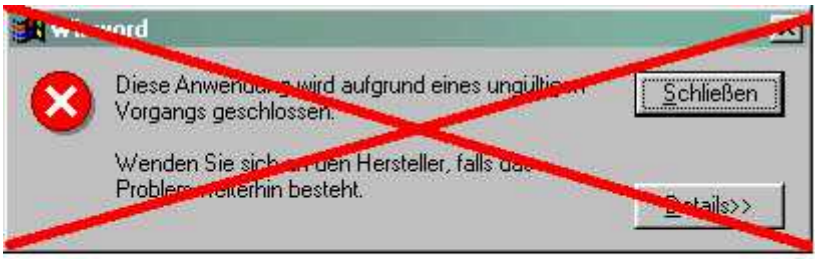
Nächster Schritt

▼ Rechner im abgesicherten Windows-Modus starten

Unbrauchbare Fehlermeldungen

▲ Zurück zum Schema

Leider werden hilfsbereite Personen z.B. in Foren oder in News-Gruppen aber auch am Stammtisch in der Strassenbahn oder im Freiluftbad mit unbrauchbaren Fehlermeldungen konfrontiert zu denen die Geplagten dann etwas aussagen sollten, das sie ja Informatiker oder sonst so was sind.



Das Nachtgebet des Wölkchenprogramms von Microsoft ist unbrauchbar, das es – wohl erkennbar – nichts aussagt!

Erst nach dem Klick auf die Schaltfläche «DETAILS» erscheint die relevante Information.

Hier die hauptsächlichsten Fehler, welche von vielen Hilfesuchenden begangen werden:

- Fehlermeldungen werden nicht im Wortlaut sondern so ungefähr aus dem Gedächtnis wieder gegeben.
- Es werden Fehlermeldungen zitiert, welche nichts ausser das Nachtgebet von Microsoft beinhalten. (Siehe dazu die einblendete Grafik.)
- Fehlermeldungen von *Dr. Watson*. Siehe dazu das Kapitel Fehlermeldungen von Dr. Watson, gleich anschliessend.

Dr. Watson Fehlermeldungen

Unter **Windows NT** werden Programmabstürze meistens von einer Art Witzfigur namens «*Dr. Watson*» abgefangen und begleitet. Auf Ihrem Bildschirm erscheint zwar auch dann eine Mendlungsbox allerdings mit einer nichtsaussagenden Meldung, welche immer das selbe Muster aufweist und etwa wie folgt aussieht:

winword.exe Ausnahme: (0xc000008c), Adresse: 0x6500631e

Diese Meldung ist wertlos!

Trotz Doktortitel, Latein und Einsatz eines schlechten *Walt Disney* Plagiats seitens Microsoft: Dr. Watson Meldungen sind für eine Ferndiagnose absolut wertlos!

Allerdings führt Windows NT eine Protokolldatei über Abschmierereien. In diesem Protokoll sollten Sie eine Fehlermeldung finden, welche mehr aussagt. Insbesondere der Name des **Moduls**, welches den Fehler verursachte, ist dabei ja wichtig. Im übrigen wird auch einen *Dump* gezogen. Bei einem *Dump* handelt es sich um eine Aufzeichnung des Adressraumes - (das ist der Arbeitsspeicher, welches ein bestimmtes Programm bzw. eine Instanz davon belegt) - während der Fehler auftrat. Ein *Dump* eignet sich nicht zur Ferndiagnose, ganz abgesehen davon, dass es dafür Spezialisten benötigt, welche zudem Ihr Umfeld sehr genau kennen müssen.

Da ich davon ausgehe, dass Windows NT vorwiegend in Betrieben eingesetzt wird, schlage ich vor, beraten Sie sich mit Ihrem **Netzwerkadministrator**. Diese Person, wird bestimmt in der Lage sein, die relevante Fehlermeldung für Sie heraus zu suchen.

Anmerkung

☹ Unverständlich und sehr ärgerlich für mich ist es, dass die Firma Microsoft es nicht schafft, ihre Fehlermeldung endlich zu ennumerieren, also jeder Fehlermeldung eine Nummer oder ein Code zu vergeben, welche anwendungsübergreifend eindeutig ist.. Im Grossrechnerbereich ist dies seit Jahren eine Selbstverständlichkeit und ermöglicht erst, die systematische Analyse von Problemen.

Rechner im abgesicherten Windows-Modus starten

▲ Zurück zum Schema

Um sicher zu stellen, dass der Fehler nicht von einem fehlerhaften Gerätetreiber oder einer Funktion, welche im Hintergrund abläuft (Hintergrundprogramm welches beim Start von Windows automatisch aufgerufen wird) verursacht wird, müssen Sie den Rechner im abgesicherten Windows-Modus starten.

Gehen Sie im Einzelnen wie folgt vor:

Windows 95, Windows 98 und Windows ME

1. Sichern Sie Ihre Arbeit.
2. Führen Sie den Vorgang ***Start - Beenden - Neu starten*** aus.
3. Der Rechner fährt nun kurz herunter und startet anschliessend gleich wieder auf.
4. Achten Sie nun ganz genau auf die Meldungen, welche beim Hochfahren erscheinen. Sowie der Meldungsfluss welches das BIOS generiert vorbei ist und bevor Windows startet, betätigen Sie die Taste **[Strg]** **Windows 98** bzw. die Taste **[F8]** **Windows 95 und Windows ME** und halten diese gedrückt.
5. Der Rechner sollte jetzt, noch auf DOS-Ebene, ein Menü Anzeigen.
6. Wählen Sie den Punkt: **Abgesicherter Modus**.
7. **Anmerkung:** Der Zeitpunkt, wenn Sie die erwähnte Taste drücken sollten ist wichtig. Wiederholen Sie den beschriebenen Vorgang allenfalls mehrmals, sollte es beim ersten Mal nicht klappen.

Windows2000

1. Klicken Sie auf **Start** und dann auf **Beenden**.
2. Klicken Sie auf **Neu starten** und dann auf **OK**.
3. Drücken Sie die Taste **[F8]**, wenn die Meldung «*Wählen Sie das zu startende Betriebssystem*» angezeigt wird.
4. Markieren Sie mit den Pfeiltasten die entsprechende Option für den abgesicherten Modus, und drücken Sie die **EINGABETASTE**.

Windows NT

▲ Zurück zum Schema

Unter diesem Betriebssystem steht leider kein *abgesicherter Modus* zur Verfügung. Es gibt aber die eine oder andere Alternative dazu. Kontaktieren Sie Ihren Netzwerkadministrator, um mehr darüber zu erfahren.

Anmerkung

- Der abgesicherte Windows-Modus ist reduziert. Sie werden feststellen, dass der Bildschirm unangenehm flackert, der Drucker nicht ansprechbar ist, Ihr Modem nicht funktioniert u.v.a.m. Es geht aber nicht darum, dass Sie lange unter diesen Bedingungen arbeiten sollen; es geht viel mehr darum, negative Einflüsse von z.T. hochkomplexen Treibern auszuschliessen.

Was ist zu tun, falls der Fehler auch im abgesicherten Windows-Modus auftritt?

In diesem Falle, fahren Sie mit der nächsten Aktion in der Tabelle fort.

Was ist zu tun, falls der Fehler im abgesicherten Windows-Modus nicht auftritt?

In diesem Falle, ist wohl nicht MS Word der direkte Verursacher des Fehlers. Sie sollten nun Ihr Augenmerk auf die **Gerätetreiber** und auf die **Hindergrundprogramme** richten. Ob Sie mit dem ersteren oder dem letzteren beginnen, ist in den meisten Fällen Geschmacksache.

Die Rolle der Gerätetreiber

Die beiden Treiber, welche die Funktionalität von MS Word stören können (und dies in der Praxis auch recht oft tun) ist einerseits der Grafikkartentreiber und andererseits der Druckertreiber. Weitere Treiber auf Ihrem Rechner haben in diesem Zusammenhang eher eine marginale Bedeutung.

Die zentrale Rolle des Druckertreibers beim Arbeiten mit MS Word

Aber: unterschätzen Sie die Rolle des **Druckertreibers** nie Mals!. Ich kenne keine Anwendung, welche so eng mit dem Druckertreiber zusammenarbeitet, wie MS Word. Auch wenn Sie sich keinen Zusammenhang vorstellen können: Sie tun sich keinen Gefallen, wenn Sie den Druckertreiber von vorn herein als Verursacher des Problems ausschliessen. Dies gilt insbesondere dann, wenn der Fehler in der Seitenlayout-Ansicht auftritt. Steht das Problem im Zusammenhang mit der Seitenansicht oder mit Schriften, dann ist es sogar hoch wahrscheinlich, dass dies mit dem Druckertreiber in Verbindung steht.

Home Page des Geräteherstellers aufsuchen

Begeben Sie sich also auf die *home page* des Hersteller Ihres Druckers oder Ihrer Grafikkarte und suchen Sie dort die Support-Ecke auf. Ich bin sicher, dass es in dieser Ecke auch eine Kategorie mit dem Titel **Treiber** (englisch übrigens *Driver*) gibt. Achten Sie darauf, dass der Treiber exakt für Ihr **Model** gilt und achten Sie weiter darauf, dass Sie den Treiber für das von Ihnen verwendete **Betriebssystem** aufstöbern.

Aktuelle Treiber für ein Gerät herunter laden

Laden Sie also diesen Treiber bzw. das Paket herunter. Beachten Sie anschliessend die Anweisung des Herstellers. Oft richtet es die Herstellerfirma so ein, dass Sie den Treiber nach dem Download auf Doppelklick installieren können.

Feststellen, welche Grafikkarte zum Einsatz kommt

Wogegen die meisten Leute wissen, welches Druckermodel eingesetzt wird, ist dies bei Grafikkarten aus verständlichen Gründen nicht immer der Fall, ist diese doch fest im PC fest eingebaut. Es gibt allerdings einige Methoden, um dieses Ding, welches in der Regel unmittelbar hinter dem Monitoranschluss - in Richtung Rechner eingebaut ist, zu identifizieren.

Falls der Händler, welche Ihnen den PC verkauft hat, seine Sache gut gemacht hat, müsste dies aus den Unterlagen, welche Ihnen bei der Auslieferung übergeben wurden, klar deklariert sein.

Wenn diese Unterlagen aus irgend einem Grund nicht zur Verfügung stehen, dann können Ihnen in vielen Fällen die Meldungen, welche auftauchen, wenn Sie den Rechner hoch fahren (*booten*) auf die Sprünge helfen. Moderne Grafikkarten identifizieren sich nämlich selbst. Da ein PC solange die Grafikkarte nicht initialisiert wurde noch gar nichts anzeigen kann, ist es oft die aller erste Meldung, welche auf dem Bildschirm nach dem Einschalten des Rechners erscheint. Sollte der Meldungsfluss zu schnell ablaufen, was oft der Fall ist, dann können Sie im richtigen Moment die Pause-Taste auf Ihrer Tastatur drücken und Name und Modell der Grafikkarte in aller Ruhe vom Bildschirm abschreiben.

Eine weiter Quelle um an diese Informationen heran zu kommen ist die Systemsteuerung zu der Sie Zugriff haben nachdem Windows aufgestartet ist. Gehen Sie im einzelnen wie folgt vor:

1. Klicken Sie ***Start - Einstellungen - Systemsteuerung*** .
2. Führen Sie einen Doppelklick auf das Symbol mit der Bezeichnung **System** aus.
3. Wechseln Sie in den Reiter **Geräte-Manager**.
4. Suchen Sie nun in der Auflistung nach dem Eintrag **Grafikkarte**.
5. Erweitern Sie die Auswahl, womit auch schon die gewünschte Information erscheint.

Nächster Schritt

▼ Feststellen, ob ein Konflikt mit einem Hintergrund Programm besteht

Feststellen, ob ein Konflikt mit einem Hintergrundprogramm besteht

▲ Zurück zum Schema

Immer noch unter der Annahme, dass der Fehler, welche Sie plagt zwar im Vollmodus auftritt im abgesicherten Windows-Modus aber nicht, ist es neben der Überprüfung der Treibersoftware nötig, Konflikte mit laufenden Hintergrundprogramme allenfalls zu erkennen und zu eliminieren.

Dazu starten Sie Ihren Rechner neu und zwar im Vollmodus (also auf die übliche Art und Weise). Anschliessend lassen Sie sich den **Task-Manager** anzeigen und beenden forciert alle Programme, welche für den Betrieb des Rechners und der Word-Anwendung nicht unbedingt notwendig sind. Gehen Sie im Einzelnen wie folgt vor:

1. Drücken Sie die Tastenkombination **[Strg + Alt + Lösch]**. Damit wird ein Dialog mit der Bezeichnung **Anwendung schliessen** angezeigt. Dieser Dialog wird sich immer in den Vordergrund drängen.
2. Markieren Sie nun also eine Task nach der anderen und betätigen Sie dann die Schaltfläche **Task beenden** **Achtung:** Die einzigen beiden Tasks, welche Sie **nicht** beenden dürfen nennen sich **Explorer** und **Systray**.
3. Warten Sie nun bis die markierte Task von Windows beendet wurde und fahren Sie dann mit der nächsten Task fort, unter Berücksichtigung, dass die zwei genannten Tasks nicht beendet werden dürfen
4. Wenn nur noch die zwei erwähnten Tasks aktiv sind, starten Sie Word und überprüfen, ob der Fehler in Word in dieser neuen Umgebung immer noch auftritt

Sollte das selbe Problem immer noch in Word auftreten, dann ist ein Konflikt mit den Hintergrundprogrammen auszuschliessen,; sie müssen den Fehler somit an einem anderen Ort suchen.

Tritt allerdings das Problem jetzt nicht mehr auf, dann sollten Sie den Rechner neu starten und die Übung wiederholen. Diesmal sollten Sie aber nach jeder Task, welche sie beenden, prüfen, ob der Fehler in Word verschwunden ist. Sobald der Fehler verschwunden ist, haben Sie die Task, welche den Konflikt auslöst erfolgreich identifiziert.

Verzichten Sie zukünftig auf diese Task, welche Word das Leben schwer macht, in dem Sie das entsprechende Programm ordnungsgemäss deinstallieren oder nehmen Sie Kontakt mit dem Hersteller dieser Software auf.

Nächster Schritt

▲ Punkt Nr. 3 im Schema

Bausteine aus der defekten «*Normal.dot*» zurück gewinnen

▲ Zurück zum Schema

Sie haben sich also - wie vorgeschlagen - entschieden, die Datei mit dem Namen «Normal.dot» nach «Alt.dot» umzubenennen. Beim nächsten Start hat Word dann eine frische Standardvorlage generiert.

Es geht nun darum, die Bausteine, welche Sie im Laufe der Zeit erstellt haben, von der «Alt.dot» in die neue «Normal.dot» zu übernehmen. Dies betrifft im Einzelnen folgende Komponenten:

Bausteine, welche zurückgewonnen werden können

- Formatvorlagen
- AutoTexte
- Benutzerdefinierte Symbolleisten
- Makroelemente wie Module, UserForms und Klassenmodule

Um diese Komponenten zu retten, gehen Sie wie folgt vor:

1. Starten Sie Word
2. Rufen Sie vom Hauptmenü den Dialog ***Extras - Vorlagen und Add-Ins... - Organisieren...*** auf. Dieser Dialog setzt sich aus vier Registern zusammen, namentlich: Formatvorlagen - AutoTexte - Symbolleisten und Makros.
3. Klicken Sie auf der linken Hälfte des Dialoges den Schaltknopf mit der Bezeichnung **Datei - Schliessen...**. Dabei ändert sich die Beschriftung dieses Schaltknopfes in *Datei Öffnen...*.
4. Klicken Sie nun also auf **Datei öffnen...** und wählen Sie dann die Dokumentenvorlage (DOT) mit dem Namen **Alt.dot** aus und bestätigen Sie Ihre Wahl.
5. Vergewissern Sie sich, dass auf der rechten Seite des Dialoges die Dokumentenvorlage mit dem Namen **Normal.dot** geöffnet ist. Allenfalls stellen Sie dies bitte noch sicher.
6. Mittels der Schaltfläche **Kopieren** können Sie nun z.B. die Formatvorlagen von der *Alt.dot* nach der *Normal.dot* kopieren.
7. Verfahren Sie mit AutoTexten, benutzerdefinierten Symbolleisten und Makroelemente ebenso.

Bausteine und Einstellungen, welche nicht bzw. nur umständlich zurückgewonnen werden können

- **Word-integrierte Symbolleisten**, welche Sie verändert haben (die benutzerdefinierten Symbolleisten stellen dagegen - wie bereits erwähnt - kein Problem dar.)
- **Benutzerdefinierte Tastenkürzel** («*Shortcuts*»), welche über *Extras - Anpassen - Tastatur* eingebracht wurden.
- Gewisse (speziell formatierte) AutoKorrektur-Einträge
- Einstellungen, welche die **Silbentrennung** betreffen.
- Einstellungen, welche den **Zoom-Faktor** betreffen
- Erstellte Verweise (*VBA-Editor - Extras - Verweise...*).

Hinweis

Sie ärgern sich auch, dass die «*Normal.dot*» so oft kaputt geht und das Microsoft es bisher nicht zu Stande gebracht hat, eine Mimik zu bieten, welche diese für MS Word so vitale Datei z.B. bei jedem Start von Word vielleicht in komprimierter Form sichert?

Wenn Sie mindestens drei ;-) dieser Fragen mit Ja beantworten können, dann senden Sie doch bitte ein E-Mail mit diesem Wunsch an folgende existierende Adresse: mailto:mswish@microsoft.com

«*Word-Data*» Schlüssel in der Registrierdatenbank umbenennen bzw. löschen

▲ Zurück zum Schema

Versierte Benutzer

Schliessen Sie Word und nenne Sie folgenden Schlüssel (und zwar den ganzen Ast) in der Registrierdatenbank nach *Data1* um:

HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data
Word 97 bzw.

HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Data
Word2000

Übrige Benutzer

Gehen Sie bitte wie folg vor:

1. **Schliessen** Sie Word.
2. Starten Sie das kleine Programm mit dem Namen **Notepad** (*Start – Programme – Zubehör*).
3. Kopieren Sie die folgenden Zeilen (sie gelten sowohl für Word 97 wie auch für Word2000) in das leere Fenster.
4. Speichern Sie nun die Datei unter dem Namen **WData.reg** ab, wobei eigentlich nur die Dateierweiterung **reg** wesentlich ist.
5. Führen Sie eine Doppelklick auf diese Datei aus.
6. Sie werden nun nach einer Bestätigung gefragt. Bejahen Sie also die Aktion.

```
REGEDIT4

[-HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Data]
[-HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Data]
```

(Diese drei Zeilen speichern Sie unter dem Namen WData.reg ab und führen Sie dann aus.**)**

Anmerkungen

- Durch das Löschen des genannten Schlüssels gehen einige Einstellungen, welche Sie allenfalls dann neu vornehmen müssen, verloren. Der Schaden hält sich allerdings in einem recht bescheidenem Umfang, da die meisten Einstellungen in der «*Normal.dot*» gespeichert werden.

Hinweis

Sie ärgern sich auch, dass dieser Schlüssel in der Registrierdatenbank so oft kaputt geht und dass Microsoft es bisher nicht zu Stande gebracht hat, eine Mimik zu bieten, welche diese für MS Word so vitale Einstellungen z.B. bei jedem Start von Word sichert? Falls Word beim nächsten Start feststellt, dass dieser Schlüssel defekt ist, könnte er dann automatisch zurück geladen werden.

Wenn Sie mindestens drei ;-) dieser Fragen mit Ja beantworten können, dann senden Sie doch bitte ein E-Mail mit diesem Wunsch an folgende existierende Adresse: mswish@microsoft.com

Nächster Schritt

▼ Umgang mit defekten Dokumenten

Der Umgang mit defekten Dokumenten

▲ Zurück zum Schema der spezifischen Fehler
▲ Zurück zum Schema der unspezifischen Fehler
▲ Zurück zum Anfang

Wann ist eine Datei defekt?

Spätestens dann, wenn ein und die selbe Datei MS Word bei mehreren Rechner zum Abstürzen bringt, wogegen sich andere Dokumente problemlos ansehen und überarbeiten lassen, müssen Sie fest damit rechnen, dass die Datei defekt ist, und nicht mehr mit Word geöffnet oder bearbeitet werden kann. Vielleicht trifft diese Aussage auch nur partial zu.

Etwas abstrakter ausgedrückt, eine Datei ist dann defekt, wenn diese von der Anwendung, welche ihr zugeordnet ist, nicht mehr oder nur noch teilweise interpretiert (angesehen und bearbeitet) werden kann. Es kann sein, dass dabei nur eine verhältnismässig kleine Sequenz betroffen ist, angesichts der Tatsache, dass Datenverarbeitung aber eine präzise Wissenschaft ist und eingedenk, dass das interne Format von Dateien das sichtbare Format in der Regel was die Komplexität betrifft weit übersteigt, kann dies bereits katastrophale Folgen haben.

Feststellen, was für eine Datei man vor sich hat

! Mit Dateien verhält es sich auch nicht anders, als mit anderen Sachen: man sollte sie nie von der Strasse auflesen! Erkundigen Sie sich als beim Autor, um was für eine Datei es sich handelt und in welcher Version die Datei vorliegt.

Berücksichtig man alle drei Betriebssysteme für welche MS Word geschaffen wurde (Windows, Mac, MS-DOS), dann kommt man auf nicht weniger als **dreizehn Word-Versionen**. Sie müssten also wissen, um was für eine Datei es sich handelt, wenn man auch ergänzend bemerken kann, dass die Wordanwendung oft auch mit älteren Word-Versionen zu recht kommt. Dies hängt unter anderem davon ab, welche Filter Sie installiert haben.

Bedenken Sie auch, dass es kein grosses Kunststück ist, zum Beispiel eine GIF-Grafik-Datei so umzubenennen, dass diese dann mit der Dateierweiterung **DOC** daherkommt. Eine solche Datei werden Sie in MS Word auch wenn Sie Kopfstände vorführen, niemals öffnen können!

Bedeutung der Datensicherung

Wenn sie festgestellt haben, dass ein Dokument defekt ist,, dann ist es wohl in den allermeisten Fällen das Günstigste, diese Datei von der letzten Datensicherung zurück zu holen. Ich hoffe für Sie, dass eine Datensicherungsgeneration gefunden wird, auf der diese Datei noch im intakten Zustand vorliegt. Dies gilt auch eingedenk der Tatsache, dass typischerweise bis ein Tageswerk verloren geht, weil die intakte Datei nicht ganz den aktuellen Stand der defekten Datei hat. Bedenken Sie, dass angeschlagene Dateien eigentlich nur Ärger bringen können. Dies trifft generell zu, wenn auch nicht immer im selben Aussmass.

Sollten Sie bisher das periodische und professionelle Sichern von Daten vernachlässigt haben, dann ist der Zeitpunkt jetzt günstig, einen guten Vorsatz zu treffen und diesen Vorsatz auch umzusetzen.

Denken Sie daran, dass für eine professionelle Datensicherung mehrere Generationen von Nöten sind. Sie sollten als z.B. auf einen Satz zurückgreifen können, welcher vor einem Tag entstand, einem weiteren der vor einer Woche entstand, u.s.w. Diese Notwendigkeit ergibt sich daraus, dass es eine Illusion ist zu glauben, sie würden in jedem Falle binnen vierundzwanzig Stunden, erkennen, dass eine Datei defekt ist. Eine gesicherte Datei, welche aber bereits defekt ist, hilft Ihnen gar nichts.

Datensicherung ist ganz allgemein in der Informatik sehr wichtig und wenn man mit instabilen Programmen, wie MS Word arbeitet, dann um so mehr.

Grundproblematik von «geflickten» Dateien

Sind Sie trotz allem auf die defekte Datei angewiesen, dann muss zuerst unterschieden werden, ob Word bereits beim Öffnen der Datei Problem bekommt, oder erst später bei einer allfälligen Überarbeitung. Wie Sie erraten können, ist der zweite Fall etwas günstiger, allerdings immer noch mehr als ärgerlich. Bedenken Sie dabei, dass eine Datei, welche scheinbar geflickt werden konnte vielleicht doch nicht ganz integer ist und zu einem späteren Zeitpunkt wieder Probleme macht. Der Beweis, dass die Datei nun wieder völlig in Ordnung ist, können Sie jedenfalls nicht erbringen. Die Praxis wird dies mit der Zeit weisen bzw. widerlegen.

Wie kann eine Datei, welche sich mit MS Word nicht einmal mehr öffnen lässt, «repariert» werden?

Realistisch gesehen gar nicht. Das Format, wie es von Word intern gebraucht wird, ist so etwas von komplex, dass es von 99,99% der Leute nicht durchschaubar ist. Sollten Sie trotzdem eine solche Person treffen, welche dies allenfalls könnte, dann ist es sehr fraglich, ob sich der Aufwand lohnt. Diese Person könnte zur Lösung bei einem komplexen Fall durchaus Tage damit verbringen bis alles wieder in Ordnung ist und eine Entwarnung auch mit einem guten Gewissen abgeben kann.

Gehen Sie realistischerweise davon aus, dass die Formatierung ganz verloren ist, der pure Text sich jedoch mit etwas Arbeit ihrerseits, zurück gewinnen lässt. Diesen Text können Sie dann in ein neu erstelltes Dokument einfügen, wodurch Sie ein mehrheitlich unformatiertes aber intaktes Word-Dokument zurück erhalten.

Sie müssen nun die Zeichen- und Absatzformatierung nochmals vornehmen. Binden Sie dann Grafiken und Objekte neu in das Dokument ein. Sie können zwei Methoden anwenden, um an diesen Punkt zu gelangen.

Text aus beliebiger Datei wieder herstellen

Für die erste Methode, gehen Sie bitte wie folgt vor:

1. Starten Sie Word
2. Vom Hauptmenü wählen Sie den Dialog ***Datei - Öffnen***.
3. In diesem Dialog stellen Sie zuerst den Wert im Feld *Dateityp:* auf ***Text aus beliebiger Datei wiederherstellen***.
4. Wählen Sie nun die defekte Datei aus und bestätigen Sie Ihre Wahl mit **Öffnen**.
5. Schauen Sie sich nun in Ruhe an, was Ihnen da präsentiert wird und verwenden Sie *Ausschneiden/Einfügen*, um die brauchbaren Teile des Textes in ein neu erstelltes Word-Dokument zu kopieren

Dokument mit WordPad öffnen

Testen Sie doch auch noch die zweite Methode um entscheiden zu können, welcher Weg zu einem besseren Resultat führt. Gehen Sie dabei wie folgt vor:

1. Klicken Sie ***Start - Programme - Zubehör - WordPad***.
2. Öffnen Sie die defekte Datei über den Dialog ***Datei - Öffnen***.
3. Schauen Sie nun in Ruhe an, was Ihnen da präsentiert wird und verwenden Sie *Ausschneiden/Einfügen*, um die brauchbaren Teile des Textes in ein neu erstelltes Word-Dokument zu kopieren

Wie kann eine defekte Datei, welche sich mit MS Word zwar öffnen lässt aber nicht richtig bearbeitet werden kann, «repariert» werden?

Dass sich das Dokument überhaupt noch öffnen lässt, obwohl sie sich als defekt erweist. legt die Vermutung nahe, dass der Schaden begrenzt ist. Das ist zwar noch kein Grund zum Jubeln aber die Chancen stehen 50:50 oder vielleicht noch etwas höher, dass das Dokument geläutert werden kann. Als Werkzeug dazu, stehen Ihnen grundsätzlich zwei zusätzliche Methoden zur Verfügung.

Zusätzlich deshalb, weil hier die bereits besprochenen Methoden selbstverständlich auch angewendet werden können. Wenn sich die Datei aber in Word überhaupt noch öffnen läst, würde ich das Glück zuerst anders herausfordern.

Dokument unter einem weniger sensiblen Format abspeichern

Der Vorteil der folgenden Methode liegt darin, dass nicht viel Zeit dafür benötigt wird und dass die Wahrscheinlichkeit hoch ist, dass das Dokument wieder in voller Integrität zur Verfügung steht; sozusagen geläutert. Voraussetzung dafür ist selbstverständlich, dass die Methode greift. Einige Formatverluste, im geringeren Umfang, scheinen mir unter diesen Perspektiven, tragbar. Anschliessend kann das Dokument wieder in das ursprüngliche Format konvertiert werden, was Word eigentlich automatisch erledigt.

Im Vordergrund steht dabei das RTF-Format, aber auch ältere MS Word-Formate, wie z.B. MS Word 5.5 können für das Experiment herangezogen werden. Sollte sich das Dokument als widerspenstig erweisen, kommt sogar das MS Word 2.0 Format in Frage. Gehen Sie im Einzelnen wie folgt vor:

1. Ziehen Sie eine Kopie des defekten Dokuments.
2. Öffnen Sie die diese **Kopie** mit MS Word.
3. Vom Hauptmenü wählen Sie den Dialog ***Datei - Speichern unter...***.
4. In diesem Dialog stellen Sie nun den Wert im Feld *Dateityp:* auf ***Rich Text Format***.
5. Speichern Sie das Dokument mittels Schaltknopf **Speichern** unter diesem Format.
6. Vom Hauptmenü wählen Sie abermals den Dialog ***Datei - Speichern unter...***.
7. Stellen Sie nun den Wert im Feld *Dateityp:* auf ***Word-Dokument***.
8. Speichern Sie nun das Dokument mittels Schaltknopf **Speichern** nochmals, also wieder im Word-Format. Dabei erfolgt eine Abfrage, da die Kopie nun überschrieben werden soll. Bestätigen Sie diese Abfrage mit ***Ja***.

Dokument ohne letzte Absatzmarke umkopieren

Die zweite Methode ist sehr einfach anzuwenden und führt recht oft zu einem positiven Resultat. Gehen Sie wie folgt vor:

1. Öffnen Sie das defekte Dokument.
2. Erstellen Sie nun ein neues Dokument.
3. Markieren Sie in der defekten Datei alles **ausser der allerletzten Absatzmarke**, welche im Hintergrund Unmengen von Formatierungsinformationen erhalten kann und deshalb sehr anfällig ist.
4. Mit der Tastenkombination **[Strg + C]** kopieren Sie nun das markierte in die Zwischenablage und setzen den Inhalt der Zwischenablage mit der Tastenkombination **[Strg + V]** in das neue Dokument ein.
5. Überprüfen Sie nun im neuen Dokument, ob der Defekt behoben ist, und formatieren Sie dann das Dokument nach.

Programme um defekte Dateien wieder herzustellen («*Recovery*» Tools»)

Vor allem im Internet findet man zahlreiche Produkte von Drittherstellern, mit denen es möglich sein soll, defekte Word-Dateien bzw. auch andere MS Office-Dateien zu reparieren.

Diese Tools sind in der Regel kostenpflichtig und nicht gerade billig. Über ihre Tauglichkeit kann ich leider keine schlüssigen Angaben machen, da mir die entsprechende Erfahrung fehlt. Aus theoretischen Überlegungen möchte ich dazu aber doch folgendes darlegen:

Meiner Meinung nach ist es so gut wie unmöglich ein Tool zu schaffen, welches das Wiederherstellen einer defekten Word-Datei garantiert, oder die meisten der denkbaren Fälle abdecken könnte. Diese Aussage spricht aber noch nicht generell gegen solche Programme. Es ist anzunehmen, dass ein grosser Teil der Defekte, welche in der Praxis eben sehr häufig geschehen, empirisch erfasst werden können. Angenommen, die Schaffer dieser Tools, machen sich nun daran, genau diese Fehler, welche mit grosser Häufigkeit beobachtet werden zu bekämpfen, dann kann sehr wohl ein taugliches Mittel entstehen. Unter dieser Annahme und unter der Annahme, dass der potentielle Kunde mit ehrlich Aussagen von diesen Herstellern umworben wird, gibt es nichts an solchen Produkten auszusetzen.

Etwas stutzig dagegen macht mich, dass Microsoft kein entsprechendes Tool anbietet. Microsoft hat ja nun weder ein ideelles noch ein kommerzielles Interesse daran, Kunden mit defekten Dateien im Regen stehen zu lassen. Vielleicht ist die Entwicklung eines solchen *Recovery Tools* also doch an den Grenzen des Machbaren, womit man Versprechungen in dieser Richtung von Dritten mit Vorbehalt. geniessen sollte.

Am Ende dieses Skriptes finden Sie einen Link zu einem Vertreiber eines solchen *Recovery Tools*, welcher mir speziell aufgefallen ist. Wer wirklich dahinter steckt, ist mir nicht ganz klar geworden. Ich habe zwei Mal über ein e-Mail angefragt,, welcher Hersteller hinter dem Produkt steht. Diese in englisch verfassten Anfragen blieben aber unbeantwortet.

Ursachen von defekten Dokumenten

Folgende Gründe sind in Praxis hauptsächlich für zerstörte Word-Dokumente verantwortlich:

- Der Benutzer öffnet ein Dokument von der Diskette und bearbeitet es dann. Sie sollten dies nie so angehen, denn es ist einfach zu gefährlich! (Siehe dazu die Bemerkung, gleich anschliessend).
- Fehlerhafte Disketten oder unsachgemässe Lagerung von Disketten.
- Unsachgemässes «*schliessen*» von Dokumenten oder unsachgemässes herunter fahren des Rechners.
- MS Word selbst, welches in Tat und Wahrheit eine sehr instabile Anwendung ist, hat die Datei zerstört.
- Das Betriebssystem, welches ebenfalls Mängel im Bezug auf Stabilität aufweist, hat das Dokument zerstört.
- Allfällige schwerwiegende Fehler in den Programmkomponenten von MS Word.
- Computerviren

! Word-Dokument ab Diskette zu bearbeiten, ist hoch gefährlich. Das liegt unter anderem daran, dass MS Word temporäre Dateien, auch auf der Diskette ablegt. Statt von der Diskette zu arbeiten, gehen Sie besser wie folgt vor:

- Dokument von der Diskette auf die Festplatte kopieren
- Dokument von der Festplatte öffnen und überarbeiten
- Dokument speichern und schliessen; dann von der Festplatte zurück auf die Diskette kopieren

☛ Statt das anfällige Medium Diskette zu benutzen und Dateien physisch zu transportieren (was ebenfalls Risiken birgt), fragen Sie besser Ihren Internet-Provider an, ob er Ihnen Speicherplatz zu Verfügung stellt. Sie können Ihre Dateien dann hochladen und haben weltweit darauf Zugriff. Viele Provider bieten in einem begrenzten Umfang diese Dienstleistung ohne Aufpreis an.

Nächster Schritt

▲ Punkt Nr. 6 im Schema

Office2000 reparieren

▲ Zurück zum Schema

Um die Reparaturfunktion in Office2000 zu nutzen, gehen Sie bitte wie folgt vor:

1. Schliessen Sie alle Anwendungen.
2. Schmeissen Sie Ihre Office2000 CD in das Laufwerk. Falls Ihre Office-Ausgabe mehr als eine CD umfassen sollte, vergewissern Sie sich, dass es die erste CD der Sammlung ist.
3. Rufen Sie den Dialog ***Start - Einstellungen - Systemsteuerung - Software*** auf.
4. Vergewissern Sie sich, dass Sie sich im Reiter *Installieren/Deinstallieren* befinden.
5. Suchen Sie in der Liste nach einem Eintrag **Microsoft Office2000 ....**
6. Klicken Sie auf die Schaltfläche **Hinzufügen/Entfernen**.
7. Im Dialog mit der Bezeichnung Microsoft Office2000 Wartungsmodus klicken Sie auf die Schaltfläche **Office reparieren**.
8. Im folgenden Dialog wählen Sie die Option **Fehler in meiner Office-Installation reparieren**.
9. Betätigen Sie nun die Schaltfläche **Fertigstellen**.

Dieser Vorgang wird nun einige Zeit beanspruchen, wobei Sie den Fortschritt an einer Leiste mit verfolgen können.

Office 97 neu installieren

▲ Zurück zum Schema

1. Schließen Sie alle Office-Programme!
2. Schmeissen Sie Ihre Office 97 CD in das Laufwerk.
3. Klicken Sie auf **Start**, wählen Sie **Einstellungen** und klicken Sie dann **Systemsteuerung** an.
4. In der **Systemsteuerung** klicken Sie bitte doppelt auf das Symbol **Software**.
5. Im Reiter **Installieren/Deinstallieren** wählen Sie **Microsoft Office 97** durch Anklicken aus.
6. Dann klicken Sie auf **Hinzufügen/Entfernen**.
7. Wählen Sie die Option **Alles Entfernen** und folgen Sie den Anweisungen im Setup-Dialogfeld.
8. Starten Sie Ihren Rechner neu
9. Klicken Sie auf **Start**, wählen Sie **Einstellungen** und klicken Sie dann **Systemsteuerung** an.
10. In der **Systemsteuerung** klicken Sie bitte doppelt auf das Symbol **Software**.
11. Im Reiter **Installieren/Deinstallieren** wählen Sie **Microsoft Office 97** durch Anklicken aus.
12. Dann klicken Sie auf **Hinzufügen/Entfernen**.
13. Folgen Sie bitte den Anweisungen im Setup-Dialogfeld.
14. Starten Sie Ihren Rechner neu

Nächster Schritt

▼ Word-Options Schlüssel in der Registrierdatenbank umbenennen

«Word-Options» Schlüssel in der Registrierdatenbank umbenennen

▲ Zurück zum Schema

Versierte Benutzer

Schliessen Sie Word und nenne Sie folgenden Schlüssel in der Registrierdatenbank nach *Options1* um:

`HKEY_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Options`
Word 97 bzw.

`HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Options`
Word2000

Übrige Benutzer

Überspringen Sie diesen Punkt und fahren Sie mit der nächsten Aktion fort. Diese Sache hat keine hohe Priorität.

Nächster Schritt

▼ AutoKorrektur-Datein und Benutzerwörterbücher umbenennen

AutoKorrektur-Dateien und Benutzerwörterbücher umbenennen

▲ Zurück zum Schema

AutoKorrektur-Dateien

Um die AutoKorrektur-Dateien umzubenennen gehen sie bitte wie folgt vor:

1. Rufen Sie den Dialog ***Start - Suchen - Dateien/Ordner...*** auf.
2. Tippen Sie im Feld *Name:* folgende Zeichenfolge ein: ***.acl**.
3. Stellen Sie im Feld *Suchen in:* den Wert **Lokale Festplatten** ein.
4. Klicken Sie nun auf den Schaltknopf **Starten**.
5. Wenn die Suche beendet ist, markieren Sie den ersten gefundenen Eintrag, betätigen die Taste **[F2]** und fügen dem Dateinamen das Word «alt» bei, also z.B. aus *«Benutzer.acl»* soll **«Benutzer alt.acl»** werden.
6. Verfahren Sie für die anderen gefunden Dateien ebenso. Möglicherweise wird auch nur eine einzige Datei gefunden.

Benutzerwörterbücher

Um die Benutzerwörterbücher umzubenennen gehen sie wie bei den AutoKorrektur-Dateien vor. Lassen Sie aber diesmal nach der Zeichenfolge ***.dic** suchen.

Nächste Schritte

▼ Hilfe von der Microsoft Online-Datenbank
▼ Empfehlenswerte Foren und News-Gruppen

Hilfe von der Microsoft Online-Datenbank

▲ Zurück zum Schema

Quellen

Microsoft bietet im Internet verschiedene Portale an, um bei einem technischen Problem auf den Erfahrungsschatz des Softwareherstellers zurückzugreifen.

Dazu sind am Ende dieses Kapitels drei Links eingeblendet. Hier eine Beschreibung der Vor- und Nachteile dieser Portale:

- **Die Microsoft *Knowledge Base* in deutscher Sprache**
  - Das Volumen (Zahl der Einträge) ist im Vergleich zur englischsprachigen Datenbank wesentlich geringer.
  - Sie können nach deutschen Begriffen suchen und die Dokumentation ist in deutsch verfasst.
  - Die Oberfläche ist einfach zu handhaben, selbsterklärend, allerdings in den Details doch etwas unpraktisch und nervig.
  - Die Suche ist träge (langsam).

- **Die Microsoft *Knowledge Base* in englischer Sprache**
  - Sie müssen die vorliegende Fehlermeldung ins englische (also eigentlich zurück ins Original) übersetzen. Dies gelingt wohl nur mit Erfahrung; nicht zuletzt deshalb da es sich nicht um Schulenglisch sondern um Fachchinesisch handelt. Allerdings die Modulnamen sind selbstverständlich in allen Sprachen die selben.
  - Das Volumen (Anzahl der Einträge) ist sehr gross.
  - Die Dokumentation ist in englisch verfasst.
  - Die Oberfläche ist einfach zu handhaben, selbsterklärend, allerdings in den Details doch etwas unpraktisch und nervig.
  - Die Suche ist träge (langsam).

- **Microsoft TechNet Insights and Answers for IT Professionals**
  - Sie müssen die vorliegende Fehlermeldung ins englische (also eigentlich zurück ins Original) übersetzen. Dies gelingt wohl nur mit Erfahrung; nicht zuletzt deshalb da es sich nicht um Schulenglisch sondern um Fachchinesisch handelt. Allerdings die Modulnamen sind selbstverständlich in allen Sprachen die selben.
  - Das Volumen (Anzahl der Einträge) ist optimal, da auch Einträge von Nebenstellen, welche zu Microsoft gehören, berücksichtigt werden.
  - Die Dokumentation ist in englisch verfasst.
  - Die Oberfläche ist etwas spartanisch, dass muss aber kein Nachteil sein.
  - Die Suche ist schnell.

Registrierung

Eventuell müssen Sie sich zuerst registrieren, bevor Sie Zugang zur Suchmaschine erhalten. Dies muss grundsätzlich nur einmal geschehen. Allenfalls können Sie ausschliesslich nach Artikel zu Produkten suchen lassen, welche tatsächlich auf Ihrem Rechner installiert und lizenziert sind.

Eventuell deshalb, weil es anscheinend eine Rolle spielt, auf welchem Weg (Link) Sie zum ersten Mal Zugang zu den Datenbanken erlangen. Eventuell auch deshalb, weil Sie Ihre Funktion wählen können. Das geht dann von Heimbenutzer über professioneller Anwender bis hin zum IT-Spezialist. Um nicht bevormundet zu werden, wählen Sie die professionellste bzw. technischste Stufe.

Suchoberfläche

Bei der ersten und zweiten aufgeführten Quelle, wählen Sie in einer vorgegebenen Liste zuerst das Produkt (in unserem Falle also Word 97 oder Word2000) aus. Anschliessend wählen Sie ebenfalls aus einer vorgegebenen Liste Ihr Betriebssystem aus. In einem Feld, welches freie Texteingabe ermöglicht, geben Sie dann die Suchbegriffe ein.

Beim TechNet geht das direkter. Sie haben ein Feld, in dem Sie Suchbegriffe eingeben können. Sie können auch zwischen einem Modus **einfache Suche** und **detaillierte Suche** auswählen. Ich empfehle Ihnen die detaillierte Suche.

Suchbegriffe und Syntax

Auch hier gilt: viele Wege führen nach Rom. Ich persönlich favorisiere die «*Boolean-Suche*». Damit können Suchbegriffe logisch verknüpft werden. Im Wesentlichen stehen die beiden Operatoren **UND** und **ODER** zur Verfügung. Der Operator **UND** ist **einschränkend.** Somit finden der Ausdruck «*Haus und Kirche*» ausschliesslich Skripte in denen sowohl Haus wie auch Kirche vorkommt. Der Operator **ODER** dagegen ist **erweiternd.** Somit findet der Ausdruck *«Haus oder Kirche»* Skripte in denen entweder Haus oder Kirche vorkommen bzw. alle beiden.

Arten von Abstürzen (Akronyme)

Zwei Begriffe bzw. Akronyme möchte ich Ihnen nicht vorenthalten. Es gibt nämlich grundsätzlich zwei Arten von Abstürzen. Die einen werden von Windows in der Fehlermeldung als **Ungültige Seite** ausgewiesen. Diese nennen sich im englischen «*Invalid Page Fault*» oder kürzer **IPF**. Die anderen werden als **Allgemeine Schutzverletzung** ausgewiesen, sind wesentlich seltener zu beobachten und nennen sich in englisch «*General Protection*» *Fault* oder kürzer **GPF**.

Word-Versionen (Akronyme)

Word 97 wird als **WD97** dargestellt, wogegen Word2000 als **WD2000** dargestellt wird.

Alles unter einen Hut bringen

Somit haben wir alles zusammen, was es für eine effiziente Suche in **TechNet.** braucht.

Angenommen, Sie werden in **Word2000** von eine Absturz geplagt welche wie folgt ausgewiesen wird:

*Winword verursachte einen Fehler durch eine **ungültige Seite** im Modul **KERNEL32.DLL***

dann geben Sie folgenden *Boolean* Suchausdruck ein:

WD2000 AND IPF AND KERNEL32.DLL

Links zu den Datenbanken von Microsoft

- ► **Microsoft knowledge base in deutscher Sprache**
- ► **Microsoft knowledge base in englischer Sprache**
- ► **MicrosoftTechNet**

Empfehlenswerte Foren und News-Gruppen

▲ **Zurück zum Schema**

Wenn Sie Ihr Problem in einem Forum oder einer News-Gruppe diskutieren möchten, dann vergessen Sie nicht Ihre **Word-Version** anzugeben und allenfalls auch die **Version Ihres Betriebssystems**. Das Posten von Abstürzten der Word-Anwendung ohne Zitat der **Fehlermeldung im Wortlaut**, ist nahezu sinnlos und verursacht unnötigerweise Rückfragen, welche für alle langweilig sind. Fehlermeldungen von Dr. Watson sind ebenfalls untauglich zur Fehlerbehebung. Beschaffen Sie sich in diesem Falle die Fehlermeldung im Original. Da es diese Witzfigur, Dr. Watson, nach meinem Informationsstand ausschliesslich in **Windows NT** gibt, kontaktieren Sie zuerst Ihren Netzwerkadministrator. Anhänge jeglicher Art sind auch in News-Gruppen verpönt und eigentlich gar nicht erlaubt.

Foren

Folgendes Forum zur Diskussion von MS Word-Problemen ist sehr empfehlenswert:

- ► **Das Spotlight Forum: Online-Treff rund um MS Word**

News-Gruppen

Microsoft unterhält zahlreiche News-Gruppen, auch in deutscher Sprache, welche von vielen fachkompetenten und hilfsbereiten Leute besucht werden. Beiträge für News-Gruppen werden in Outlook Express oder vergleichbaren Anwendungen geschrieben und gelesen. Hier die zwei wichtigsten deutschsprachigen News-Gruppen im Zusammenhang mit MS Word. Wenn Sie einer der Links klicken, erfolgt die Einrichtung unter Outlook Express **vollautomatisch**.

► **news://msnews.microsoft.com/microsoft.public.de.word** (für allgemeine Fragen zu MS Word)

► **news://msnews.microsoft.com/microsoft.public.de.word.vba** (für Fragen, welche Word VBA betreffen)

Bitte respektieren Sie diese Unterscheidung.

Nächster Schritt

▼ **PC «Hot Lines»**

PC «Hot Lines»

▲ **Zurück zum Schema**

Verfügbarkeit und Kompetenz

«Hot Lines» für geplagte Anwender werden immer zahlreicher. Glücklicherweise hat auch die Kompetenz im Schnitt zugenommen. Der Service ist allerdings bestimmt nicht billig. Das ist teilweise verständlich. Man rechne doch einfach mal selbst nach und zwar unter Berücksichtigung der Tatsache, dass die Löhne für kompetente Leute auf diesem Gebiet hoch sind.

Kostenspanne

Gravierender wiegt aber, dass die Kosten schlecht im voraus abzuschätzen sind. Die Kostenspanne zur Behandlung eines Falles, wie er hier Thema ist, liegt nach einer aktuellen Untersuchung eines der grossen deutschen PC-Magazine zwischen DM 35,00 (wenn Sie sehr viel Glück haben) und DM 195,00 (wenn Sie Pech haben).

Diese Preisspanne ist so hoch, dass eine halbwegs vernünftige Kosten/Nutzen-Analyse oft gar nicht möglich ist.

Viele Service-Stellen rechnen diesen Service, welcher fast ausschliesslich fernmündlich abläuft, im Minutentakt über die Telefongebühren ab. Sie sollten sich sehr genau erkundigen, welcher Tarif hier zum tragen kommt. In der Schweiz z.B. wurde die höchst mögliche Gebühr für eine Einheit (Eine Minute) unlängst von CHF 2.13 auf über CHF 4.-- erhöht!

Ich persönlich würde nur einen Service beanspruchen, welchen ich kenne oder der mir aus zuverlässiger Quelle nahe gelegt wurde.

Die Hot Line von Microsoft

Microsoft unterhält eine kompetente Hotline bei München, welche auch für den Kundenkreis in Österreich und in der Schweiz zuständig ist. Microsoft geht allerdings bei der Verrechnung einen anderen Weg. Die Telefongebühren schlagen mit maximal DM 0,50 pro Minute zu buche. Dieser Tarif gilt auch für Kunden ausserhalb Deutschlands. Dabei handelt es sich selbstverständlich nur um die Gebühren, welche die Telekommunikationsgesellschaft einzieht. Die Beratung selbst muss nach Ablauf der Garantieleistung z.B. per Vorkasse bezahlt werden und beträgt ca. DM 160,00 pro Fall oder *«Incident»*, wie das bei Microsoft genannt wird.

Für diesen Betrag verpflichtet sich Microsoft - im Umfang des Möglichen - das Problem abschliessend zu lösen. Diese Verpflichtung hält die Firma erfahrungsgemäss ein. Jedenfalls geben sie sich Mühe. Besser machen können diese Leute die Office-Anwendungen selbstverständlich nicht. Es gibt auch die Möglichkeit bei Microsoft einen Jahresvertrag abzuschliessen. Die Leistungen sind auch dort klar geregelt, ein Freipass, dass Sie nun jeden Tag den Service beanspruchen können, ist dies selbstverständlich nicht.

Garantieleistung von Microsoft

Für Kunden, welche die **Lizenz direkt von Microsoft** haben, also für die MS Office-Suite viele Hunderte Deutsche Mark bezahlt haben, erhalten **zwei *Incidents* frei.** Für Besitzer einer **OEM-Lizenz**, also einer Lizenz welche Sie im Paket mit neuer Hardware erworben haben, gilt dieses Angebot nicht. OEM-Kunden können gar **keine Unterstützung** seitens Microsoft beanspruchen. Theoretisch könnten sich Besitzer von OEM-Lizenzen an den PC-Händler wenden. Dabei kommt in der Regel aber nicht viel Brauchbares heraus.

Sollte es sich herausstellen, dass Ihre Probleme dadurch entstanden ist, dass das Office-Produkt einen klaren Mangel aufweisst, wird Ihnen keinen *Incident* verrechnet bzw. abgezogen. In der Praxis entscheidet aber Microsoft die Schuldfrage. Somit bleibt die Hoffnung auf einen freien *Incident* Spekulation.

Kunden mit Kaufverträgen, welche Schweizerischem Recht unterliegen

Schweizer Kunden fahren etwas besser, da der Lizenzvertrag, welchen sie von Microsoft erhalten haben gar nicht den gesetzlichen Mindestanforderungen einer Garantie in der Schweiz entspricht. Wenn Sie also das Produkt in der Schweiz erworben haben, können Sie die Hotline ein ganzes Jahr lang beanspruchen ohne dass Sie zusätzlich bezahlen müssten. Nach einem Jahr erlischt dann aber jegliche Garantie. Machen Sie dieses Recht bei Microsoft geltend, falls der Kaufvertrag Schweizerischem Recht unterliegt. Microsoft wird Sie wahrscheinlich nicht auf diesen Vorteil aufmerksam machen. Die genauen Bedingungen für Österreich, sind mir leider nicht bekannt.

Zeit und Kosten sparen

Um das Telefonat so kurz wie möglich zu halten, was viel Geld sparen kann, legen Sie die **Produkte-ID** bereit. Sie finden Sie z.B. in MS Word unter dem Menüpunkt ***? - Info***. Sie sollten die Fehlermeldung im vollen Wortlaut aufgeschrieben oder noch besser ausgedruckt haben. Ihr Telefon sollte nahe beim Rechner sein, den Rechner haben Sie bereits eingeschaltet.

Falls Rückrufe nötig werden, erhalten Sie neben einer **Kundennummer** auch eine ***Incident-Nummer***. Diese beiden Informationen sollten Sie aufbewahren und bereit halten.

Ablauf der fernmündlichen Beratung

Der Ablauf des Beratungsgespräch ist routiniert und systematisch. Er ähnelt dem Vorgehen sehr stark, welches Gegenstand dieses Skriptes ist. Ärgern Sie sich also nicht, wenn Sie einige Methoden, welche hier beschrieben wurden, nochmals auf Wunsch der Hot Line durchführen müssen.

Nächster Schritt

► **Office 97/2000 möglichst vollständig entfernen**

Probleme mit ausschneiden, kopieren, einfügen

▲ **Zurück zum Schema**

Es ist bekannt, dass gewisse Software-Produkte von Drittherstellern, welche Komponenten beinhalten, die immer im Hinwegrund aktiv sind, diese Probleme verursachen können. Die Probleme äussern sich so, dass MS Word wenn man die Befehle Ausschneiden, Kopieren, Einfügen anwendet, relativ oft abschmiert. Es handelt sich dabei um einen **IPF** (Ungültige Seite).

Folgende Produkte sind als Verursacher bekannt:

- **KlickTel** (ältere Versionen) - Sie sollten die Schnellstratleiste ausschalten, damit das Problem verschwindet.
- **Exceed 6.0.1**
- **Ecco Professional 3.01**

Absturz beim Drucken und Druckverwandten Funktionen

▲ Zurück zum Schema

Sollte das ausgewiesene Modul beim Absturz mit den Zeichen «*HP*» beginnen, dann setzen Sie vermutlich ein Gerät dieses führenden Herstellers von PC-Druckern ein. In diesem Falle ist die Fehleranalyse einfach; sie sollten die Software für Ihren Drucker neu installieren oder sich auf der *Home Page* des Herstellers gleich den neusten Treiber für Ihr Gerät herunter laden und installieren.

Dies gilt natürlich auch dann, wenn Sie auf Grund des Namens des fehlerhaften Moduls einen anderen Druckerhersteller erraten können.

Mehr Auskunft, über Treiber erhalten Sie im folgenden Kapitel: **Die zentrale Rolle des Druckertreibers beim Arbeiten mit MS Word**

Links zu diesem Thema

▲ Zurück zum Anfang

Eigene Links

► **Wo speichert MS Word seine Einstellungen (Umgebungsoptionen)?**

Microsoft

Englisch

► **WD2000: Part 1: Troubleshooting Invalid Page Faults (IPFs) and Exception Errors**
► **WD2000: Part 2: Troubleshooting Invalid Page Faults (IPFs) and Exception Errors**
► **WD2000: Part 3: Troubleshooting Invalid Page Faults (IPFs) and Exception Errors**
► **OFF2000: Troubleshooting Office Kernel32.dll Errors Under Windows 98**

Anonym (typisch Internet, halt)

► **OfficeRecovery**

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/IPF.htm

René Probst, Oktober 2000

Änderungen

- 24. November 2000 - Recovery Tools

Wie kann Office 97 oder Office2000 möglichst vollständig von meinem Rechner entfernen?

◄ Zurück zum Hauptindex

Problem

> **Seit Tagen, nein fast schon seit Wochen, versuche ich die beiden Office Service Releases von Microsoft auf meinem Rechner einzuspielen und immer scheitert dieses Unterfangen an irgend einer Stelle. Die Aktualisierung von Office bricht irgendwann mit irgendeiner einer lakonischen Fehlermeldung ab, welche wenig aussagt und gedanklich auch nicht nachvollziehbar ist. So erhalte ich z.B. oft die Fehlermeldung:**
>
> ***Es ist keine Office installiert oder es sind verschiedene Versionen von Office installiert.***

Problem

Zwischenzeitlich bin ich an einem Punkt, an dem ich das ganze Office-Geraffel deinstallieren möchte, um wieder sauber von vorne beginnen zu können. Bereits habe ich versucht, alles über den Setup-Assistenten zu deinstallieren. Wenn ich nun aber wieder von vorne beginne, laufe ich spätestens beim Versuch den Service Release 2 zu installieren, wieder in das selbe Desaster. Nach einem Blick in die Registrierdatenbank traue ich dem Office-Setup nicht mehr. Trotz vermeintlich vollständiger Deinstallation von Office, tummeln sich dort noch Office-relevante Einträge zu hauf.

Kann ich auf einen zuverlässigeren Weg zurückgreifen, oder muss ich jetzt die Festplatte neu formatieren, nur weil man hier den Überblick wohl auch in Redmond verloren hat?

Antwort

Die Installationsroutine von Office taugt für dieses Unterfangen tatsächlich nicht viel. Dies scheint man sogar bei Microsoft so zu sehen. Microsoft stelle sowohl für Office 97, wie auch für Office2000 ein Werkzeug zu Verfügung, um Office sauber zu entsorgen. Sie können dieses Tool gratis aus dem Internet beziehen, denn die Entsorgungsgebühr, haben Sie bestimmt schon beim Kauf der MS Office-Anwendung bezahlt. Das Tool heisst **Eraser97** bzw. **Eraser2k**. Ob es nach dem Eraser in kürze auch einen Terminator geben wird, entzieht sich allerdings meinen Kenntnissen. Gehen Sie nach dem Download wie folgt vor:

1. Schliessen Sie alle Office-Anwendungen, auch diejenigen, welche im Hintergrund laufen.
2. Starten Sie den Office-Setup und wählen Sie die Option **Alles entfernen**.
3. Führen Sie einen Doppelklick auf die Programmdatei **Eraser97.exe** aus.
4. Folgen Sie den Anweisungen der Installationsroutine.
5. Rufen Sie über ***Start - Ausführen - Programme - Eraser97*** das Programm auf.
6. Installieren Sie Office nun neu und allenfalls die Service Releases 1 und 2

Links zu diesem Thema

Microsoft

Deutsch

► **OFF97: Wie man Microsoft Office 97 vollständig entfernt**

Englisch

► **OFF97: Utility to Completely Remove Remaining Office 97 Files and Registry Entries**

► **OFF2000: Utility to Completely Remove Remaining Office 97 Files and Registry Entries**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Eraser.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

So bringen Sie ein Makro ein und verknüpfen es mit einer Schaltknopf auf einer Symbolleiste Ihrer Wahl

◄ Zurück zum Hauptindex

Frage

> Ich habe ein Makro erhalten und möchte dieses nun einbringen und es mit einem neuen Schaltknopf auf einer Symbolleiste unter Word 97 bzw. Word2000 verbinden, damit es mit einem einfachen Klick startet.

Antwort

Überlegen Sie sich zuerst, ob das Makro bei **allen** Word-Dokumenten zum tragen kommen soll oder ausschliesslich für Dokumente, welche auf einer **bestimmten Vorlage** basieren oder aber nur für ein ganz **bestimmtes** Dokument. Entsprechend Ihrer Entscheidung, wählen Sie eine der unten aufgeführten Anweisungen aus.

Makro und Schaltknopf global (allgemein) verfügbar machen

Soll das Makro bzw. der Schaltknopf in allen Dokumenten zur Verfügung stehen, gehen Sie bitte wie folgt vor:

1. Lassen Sie nach der Vorlage mit dem Namen *Normal.dot* suchen.
2. **Öffnen** Sie die Standardvorlage mit dem Namen **Normal.dot** entweder über den Windows-Explorer und zwar ***Rechtsklick - Kontextmenü - Öffnen*** oder aber von Word aus über den Dialog ***Datei - Öffnen...***.
3. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
4. Wenn Sie dort heil angekommen sind wählen Sie vom Hauptmenü den Befehl ***Einfügen - Modul***.
5. Kopieren Sie nun den erhaltenen Code in das leere Fenster, welches sich geöffnet hat.
6. Kehren Sie nun in das Word-Dokumentenfenster zurück.
7. Rufen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen...*** auf.
8. Wechseln Sie im nun erscheinenden Dialog in der Reiter **Befehle**.
9. Markieren Sie in der linken Liste *Kategorien:* den Eintrag **Makros**.
10. Suchen Sie in der rechten Liste *Befehle:* nach dem gewünschten Makro.
11. Packen Sie dieses Makro mit dem Mauszeiger und ziehen Sie ihn auf eine Symbolleiste Ihrer Wahl.
12. Immer noch im **Anpassungsmodus**, können Sie einen Rechtsklick auf den Knopf ausüben und im nun erscheinenden Kontextmenü das Aussehen des Knopfes beeinflussen. Was Sie damit alles erreichen können, wird anschliessend noch beschrieben.
13. Verlassen Sie nun den Dialog mit der Schaltfläche ***Schliessen*** speichern sie die *Normal.dot* und verlassen Sie Word.

Makro und Schaltknopf für Dokumente, welche auf einer bestimmten Vorlage basieren, verfügbar machen

Soll das Makro bzw. der Schaltknopf in allen Dokumenten, welche auf einer bestimmten Vorlage basieren zur Verfügung stehen, gehen Sie grundsätzlich genau gleich vor, wie in der oben aufgeführten Anweisung. **Öffnen** Sie aber in diesem Fall nicht die *Normal.dot* sondern die Dokumentenvorlage Ihrer Wahl.

Makro und Schaltknopf für ein bestimmtes Dokument verfügbar machen

Soll das Makro bzw. der Schaltknopf nur für ein bestimmtes Dokument zur Verfügung stehen, dann gehen Sie nach folgender Anweisung vor:

1. Öffnen Sie das fragliche Dokument.
2. Wechseln Sie mit der Tastenkombination **[Alt + F11]** in den VBA-Editor.
3. Wenn Sie dort heil angekommen sind wählen Sie vom Hauptmenü den Befehl ***Einfügen - Modul***.
4. Kopieren Sie nun den erhaltenen Code in das leere Fenster, welches sich geöffnet hat.
5. Kehren Sie nun in das Word-Dokumentenfenster zurück.
6. Rufen Sie vom Word-Hauptmenü den Dialog ***Extras - Anpassen...*** auf .
7. Wechseln Sie im nun erscheinenden Dialog in der Reiter **Befehle**.
8. Wählen Sie im Kombinationsfeld, welches mit **Speichern in:** beschriftet ist, den Name des fraglichen Dokumentes.
9. Markieren Sie in der linken Liste *Kategorien:* den Eintrag **Makros**.
10. Suchen Sie in der rechten Liste *Befehle:* nach dem gewünschten Makro.
11. Packen Sie dieses Makro mit dem Mauszeiger und ziehen Sie ihn auf eine Symbolleiste Ihrer Wahl.
12. Immer noch im **Anpassungsmodus**, können Sie einen Rechtsklick auf den Knopf ausüben und im nun erscheinenden Kontextmenü das Aussehen des Knopfes beeinflussen. Was Sie damit alles erreichen können, wird anschliessend noch beschrieben.
13. Verlassen Sie nun den Dialog mit der Schaltfläche ***Schliessen*** und speichern sie das Dokument.

Erscheinungsbild eines Schaltknopfes beeinflussen

Im Anpassungsmodus (***Extras - Anpassen..***) können Sie einen Rechtsklick auf eine Schaltfläche ausüben und dann das Aussehen des Schaltknopfes verändern. Folgende Optionen stehen Ihnen zur Verfügung:

- Im Feld **Name** können Sie der Schaltfläche einen aussagekräftigen Namen verpassen. Falls die Schaltfläche als Menüpunkt dargestellt wird. erscheint dieser Name 1:1 so wie Sie ihn eingeben. Wenn Sie diesen Menüpunkt nicht nur mit der Maus erreichen wollen sondern auch über eine Tastenkombination **[Alt +...]**, dann verwenden Sie das **&**-Zeichen (kaufmännisches und) und zwar nach dem Zeichen, welches Teil der Tastenkombination werden soll. Dieses Zeichen wird dann unterstrichen dargestellt. Sie sollten darauf achten, dass pro Menü das selbe Zeichen nur einmal zur Anwendung kommt. Auf englisch nennt sich dieses Ding übrigens *Accelerator*. Hier ein Beispiel dazu: Sie können den Befehl mit dem Namen **S**uchen... im Menü **B**earbeiten ohne die Maus zu benützen, also mit einer Tastenkombination wie folgt ansprechen: **[Alt + B + S]**.
- Mit den Befehlen **Schaltfläche kopieren** und **Schaltfläche einfügen** können Sie ein Sinnbild zwischen verschiedenen Schaltflächen austauschen. Sie können aber auch z.B. im Word-Dokument eine Grafik einfügen, diese dann in die Zwischenablage kopieren und anschliessend mit mit Befehl Schaltfläche einfügen diese Grafik auf eine Schaltfläche Ihrer Wahl bringen. Die besten Resultate erreichen Sie, wenn diese Grafik nicht zu komplex ist. Die Skalierung der Grafik wird Ihnen dabei abgenommen. Beachten Sie, dass diese Aktionen für eine Schaltfläche, welche auf *Nur Text* gesetzt ist, nicht zur Verfügung steht.
- Mit dem Befehl **Schaltfläche zurücksetzen** können Sie das Sinnbild und andere Eigenschaften der Schaltfläche auf den Word-Standard zurücksetzen. Diese Funktion ist allerdings nur für Word-integrierte Schaltflächen von Bedeutung.
- Mit dem Befehl **Schaltfläche bearbeiten...** tut sich ein kleiner Editor auf. Sie können in diesem Editor das Sinnbild der Schaltfläche verändern oder ein eigenes kreieren.
- Mit dem Befehl **Schaltflächensymbol ändern** können Sie ein Sinnbild für die Schaltfläche auswählen. Die Auswahl ist aber doch sehr beschränkt und Sie haben keine Möglichkeit diese Auswahl zu erweitern. Der einzige Ort, an dem Sie ein eigens kreiertes Sinnbild aufbewahren können ist die Symbolleiste.
- Die nächsten vier Befehle in diesem Kontextmenü legen fest, ob die Schaltfläche ein Sinnbild oder Text enthalten soll. Dabei können Sie zwischen folgenden vier Optionen auswählen:
  - **Standard** Wenn sich der Befehl in einem Menü befindet, wird sowohl das Sinnbild, wie auch der Text angezeigt. Befindet sich der Befehl auf einer Symbolleiste wird nur das Sinnbild angezeigt.
  - **Nur Text (immer)** Gleichgültig, ob sich der Befehl in einem Menü oder in einer Symbolleiste befindet, es wird immer nur der Text angezeigt.
  - **Nur Text (in Menüs)** Befindet sich der Befehl in einem Menü dann wird nur der Text angezeigt, ist er dagegen in einer Symbolleiste wird nur das Sinnbild angezeigt.
  - **Schaltflächensymbol und Text** Gleichgültig, ob sich der Befehl in einem Menü oder in einer Symbolleiste befindet, es wird immer sowohl das Sinnbild wie auch der Text angezeigt. **Anmerkung:** Normalerweise sollten Sie mit der Einstellung **Standard** gut fahren.
- Zum Schluss finden Sie noch die Option **Gruppierung beginnen**. Damit können Sie angeben, dass vor dem Befehl, welchen Sie gerade bearbeiten ein Trennstrich eingefügt werden soll.

Anmerkungen

- Ich bin mir bewusst, dass es Methoden gibt, welche von der hier beschriebenen abweichen und ebenfalls zum gewünschten Ziel führen. Die hier beschriebene Methode mag nicht die schnellste sein, hat sich aber als zuverlässig und reproduzierbar erwiesen.
- Wie Sie zu einer Liste aller Word-integrierten Sinnbilder kommen, erfahren Sie in einem der folgenden Links.

Links zu diesem Thema

Eigene Links

► **Wie kann ich eine Liste der Word-Symbole (Sinnbilder) erstellen?**

► **Symbolleistenschaltflächen und Menüleisten selbst gemacht..**

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Button.htm

René Probst, Mai 2000

Änderungen

- --- nil ---

Der Umgang mit der Registrierdatenbank und dem Registriereditor

◄ Zurück zum Hauptindex

Bedeutung der Registrierdatenbank

Seit dem Erscheinen von Windows 95 und dem definitiven Übergang zur 32-Bit Architektur, nimmt die Registrierdatenbank eine zentrale Stellung auf den meisten Kleinrechner ein. Bei einer unqualifizierten Manipulation der Registrierdatenbank, können Sie im unglücklichsten Falle, nicht nur Word oder Office lahm legen, sondern Ihren ganzen PC.

Sicherheitsmassnahmen

Folgendes ist deshalb in Ihrem eigenen Interesse angesagt:

- Seien Sie vorsichtig, wenn Sie Änderungen in der Registrierdatenbank vornehmen. Wenn Sie unsicher sind, führen Sie Änderungen nur im Vier-Augen-Verfahren durch, was nichts anderes heisst, dass Sie bei jeder Änderung zu zweit sein sollten.
- Sichern Sie die Registrierungsdatenbank regelmässig und vor allem vor einer Aktion, bei welcher Sie sich nicht ganz sicher sind.

Zugang zur Registrierungsdatenbank

Zugang zur Registrierdatenbank erhalten Sie über den Registriereditor. Um den Registriereditor zu starten, gehen Sie wie folgt vor:

1. ***START - Ausführen***
2. Geben Sie nun in der Kommandozeile die Zeichenfolge **regedit** ein und betätigen Sie die Schaltfläche ***Ok***.

Aufbau und Handhabung der Registrierungsdatenbank

Die Registrierdatenbank präsentiert sich im Registriereditor ähnlich wie Ihre Dateien im Windows-Explorer. Die Komponenten, welche im Windows-Explorer Verzeichnisse heissen, nennen sich in der Registrierdatenbank allerdings **Schlüssel**. Jeder Schlüssel kann null, einen oder mehrerer **Unterschlüssel** aufweisen. Der Aufbau ist streng **hierarchisch**, etwa mit einem Familienstammbaum zu vergleichen. Jeder dieser Schlüssel wiederum verwaltet 0, ein oder mehrere Werte, im folgenden auch Parameters genannt. Die Schlüssel werden im linken Fenster angezeigt, die Parameters mit ihren Werten im rechten Fenster. Sobald Sie einen Schlüssel markieren, erscheinen auf der rechten Seite die Parameters mit den dazugehörigen Werten. Diese Parameter können drei mögliche Formate haben namentlich:

- **Zeichenfolge** Hier sind Werte im Klartext erlaubt. Sie erweisen sich als unproblematisch in der Handhabung und es eigentlich gibt nichts spezielle zu beachten.
- **DWORD** Es handelt sich dabei um einen Wert welcher intern immer genau vier Bytes belegt. Als gültige Werte kommen **nur Zahlen** in Frage. Standardmässig werden die Werte in **hexadezimaler Darstellung** angezeigt und gespeichert. Wenn Ihnen das Umrechnen aber schwer fällt, können Sie beim editieren eines Werte in den dezimalen Modus umschalten.
- **Binärwert** Es handelt sich hierbei um eine Art von Zeichenfolge. Allerdings werden die Werte in einer Form angezeigt und abgespeichert, wie sie üblicherweise nur vom Rechner interpretierbar sind. Daraus folgt: **Ändern oder setzen Sie niemals einen Binärwert!**

Um einen Wert zu **ändern**, führen Sie einen Doppelklick auf den fraglichen Wert im rechten Fenster aus. Sie können den Wert dann in einem kleinen Fenster, welches sich öffnet, bearbeiten.

Um einen **neuen Wert** für einen bestehenden Schlüssel aufzunehmen, führen Sie nachdem Sie den fraglichen Schlüssel in der linken Hälfte markiert habe, einen Rechtsklick auf eine leere Stelle im rechten Fenster aus. Im nun erscheinenden Kontextmenü wählen Sie zuerst ***Neu*** und dann das von Ihnen gewünschte Format (oft lautet das Format ***Zeichenfolge***). Nun können Sie dem Parameter den vorgesehenen und definitiven Namen vergeben. Als letzte Aktion führen Sie einen Doppelklick auf den neu erstellten Parameter aus und vergeben diesem den gewünschten Wert.

Wenn Sie im Registriereditor einen **neuen Unterschlüssel** hinzufügen wollen, ist es sehr wichtig zuerst den richtigen Oberschlüssel zu markieren. Führen Sie nun einen Rechtklick auf den markierten Schlüssel aus. Im nun erscheinenden Kontextmenü wählen Sie nacheinander zuerst ***Neu*** und dann ***Schlüssel***. Als nächstes vergeben Sie dem Unterschlüssel den erwünschten und definitiven Namen. Anschliessend können Sie neue Parameters für diesen Schlüssel erstellen. Auch hier gilt: die Handhabung dafür ist analog zum Windows-Explorer. Auch das **Löschen eines Schlüssels** geschieht gleich wie im Windows-Explorer.

Der Registriereditor verfügt auch über eine Suchfunktion. Sie erreichen diese über den Menübefehl: ***Bearbeiten - Suchen***. Wenn Sie noch nicht über viel Routine verfügen, achten Sie darauf, dass alle Häkchen im Bereich **Suchfunktionen** aktiviert sind. In Anbetracht, dass Schlüssel in der Registrierdatenbank sehr lang und komplex sein können, bin ich sicher, dass Sie die Suchfunktion bald nicht mehr missen möchten.

Sichern und Wiederherstellen der Registrierdatenbank

Über den Menübefehle ***Registrierung - Registrationsdatei exportieren***, können Sie eine Sicherung der ganzen *Registry* vornehmen. Falls Sie die ganze Registrierdatenbank sichern wollen achten Sie unbedingt darauf, dass Sie den obersten Eintrag im linken Fenster markiert haben - dieser Eintrag heisst **Arbeitsplatz**. Vergeben Sie der Sicherungsdatei einen aussagekräftigen Namen. Vergewissern Sie sich nochmals, dass Sie alles gesichert haben und nicht nur einzelne Äste der hierarchisch aufgebauten Datenbank. Eine Vollsicherung hat eine typische Grösse von 4-8 MB. Damit ist sie leider zu gross für eine Sicherung auf einer konventionellen Diskette.

Wie schon angedeutet, können Sie auch nur einzelne Äste dieser Datenbank sichern. Eine Sicherungsdatei erhält übrigens die Dateierweiterung ***.reg** verpasst. Zumindest solche Äste lassen sich sehr einfach zurückspielen. Es reicht nämlich einen Doppelklick auf eine solche *reg-Datei* auszuführen. Nach einer Sicherheitsabfrage beginnt Windows anschliessend mit dem Zurückspielen. Selbstverständlich können Sie dies auch über den Menüpunkt ***Registrierung - Registrationsdatei importieren...*** erreichen.

Falls Sie sich entscheiden sollten, eine von Ihnen erstellte Vollsicherung wieder einzuspielen, würde ich allerdings nicht so vorgehen (ich habe dies allerdings noch nie auf diesem Wege versucht). Wenn Sie an diesem Punkt angelangt sind, dann ziehen Sie am Besten eine Fachperson bei, welche damit Erfahrung hat. Wichtig ist in solchen Situationen, dass Sie über eine Vollsicherung verfügen. Zu mindest **Windows 98** erstellt täglich eine Sicherung aus eigenem Antrieb (fast unbemerkt beim Hochfahren des Rechners). Auch diese Tatsache kann man in seine Überlegung einbeziehen, wenn wirklich mal etwas gravierendes schief gegangen ist. Windows 98 erstellt sogar mehrere Generationen, was nichts anderes heisst, dass Sie beim Zurückspielen den Tag wählen können zu dem Sie zurück gehen möchten. Entscheidend ist, dass Sie in einer solchen Situation nicht Tage oder Wochen verstreichen lassen sollten. Jede Änderung seit der Sicherung in der Registrierdatenbank, welche Sie, Windows oder ein Anwendungsprogramm in der Zwischenzeit machen, wird nämlich beim Zurückspielen wieder überschrieben.

Eine zusätzliche Sicherheit haben Sie, wenn Sie die Anweisung nach dem Muster "Löschen Sie den Schlüssel "bla" dahingehen abwandeln, dass Sie den Schlüssel nicht löschen sondern nur umbenennen. Statt den Unterschüssel mit dem Namen "Option" zu löschen, nenne Sie ihn einfach in Option1 um.

Über den Menüeintrag Registrierung - Drucken können Sie sich auch eine Zweig der Registrierdatenbank ausdrucken und somit das Davor und Danach dokumentieren.

Es gibt eine stattlich Anzahl zusätzliche Methoden, die Registrierdatenbank zu sichern. Zwei davon seien hier noch erwähnt:

- Auf der **Windows 95**-CD befindet sich das kleine Programm **ERU.EXE**. Der Name steht für *Emergency Recovery Utility*. Neben einer Vollsicherung der Registrierdatenbank erhalten Sie auch eine boot-fähige Diskette, welche es erlaubt, den Rechner auch dann noch zu starten, wenn dies von der Festplatte aus nicht mehr möglich ist. ERU ist vielleicht die beste Lösung, wenn Sie eine Sicherung auf Disketten bevorzugen, welche Sie unabhängig von der Festplatte aufbewahren können.
- In welcher Form die Registrierdatenbank in Windows vorliegt, ist kein Geheimnis. Alles nötige dazu wird in den beiden Dateien **System.dat** und **User.dat** abgelegt. Falls Sie eine Sicherung dieser beiden Dateien zur Hand haben, können Sie im Bedarfsfall also auch den Rechner bis zum DOS-Modus hochfahren. Im DOS-Prompt entfernen Sie zuerst den Schreibschutz dieser beiden Dateien auf der Festplatte. Nun überschreiben Sie die Dateien mit der Sicherung, welche Sie vorher gemacht haben und können nun Windows hochfahren oder den Rechner neu unter Windows starten. Beachten Sie bei einer Sicherung dieser Dateien folgende Punkte:
    1. Sie müssen den Dateien System.dat und User.dat zuerst verschiedene Dateiattribute rauben, bevor Sie diese kopieren können. Insbesondere muss das Attribut **versteckt** und **schreibgeschützt** entfernt werden. Dies erreichen Sie im Windows-Explorer über Rechtsklick - Kontextmenü - Eigenschaften. Unter DOS können Sie den Befehl **Attrib** verwenden.
    2. Die Datei System.dat ist zu gross, um sie in einem Stück auf eine konventionelle Diskette zu kopieren.

URL

http://mypage.bluewin.ch/reprobst/WordFAQ/Regedit.htm

René Probst, Oktober 1999

Änderungen

- --- nil ---

Schlussbemerkungen

◄ Zurück zum Hauptindex

Programmversionen

Der hier publizierte VBA-Code wurde teilweise unter Word 97 vermehrt aber unter Word2000 entwickelt und getestet.

Feedback

Anmerkungen, Anregungen, (relevante) Korrekturen, etc. sind über folgende E-Mail-Adresse erwünscht René Probst-

Anfragen zu konkreten Word-Problemen, welche täglich in stattlicher Zahl hier eintreffen, werden nicht beantwortet, sondern wandern nach sehr flüchtiger Durchsicht in den Papierkorb. Bei dieser Durchsicht interessiert mich eigentlich lediglich, was die Leute beschäftigt und wo sie allenfalls anstehen. Themen, welche von allgemeinen Interesse sind und sich recherchieren und beschreiben lassen, haben hohe Priorität bei der Planung von Erweiterungen meiner WordFAQs. Senden Sie am Besten keine E-Mails und keine Anhänge, insbesondere keine Anhänge, welche Makros enthalten.

Hinweise für Hilfesuchende

Wenn Sie Ihre Anfrage im Spotlight-Forum oder in einer der zahlreichen Microsoft News-Gruppen posten, habe Sie dagegen gute bis sehr gute Chancen, dass sich jemand Ihrer Frage in kompetenter Weise annimmt. Vergessen Sie nicht Ihre Word-Version anzugeben und allenfalls auch die Version Ihres Betriebssystems. Das posten von Abstürzten der Word-Anwendung ohne Zitat der Fehlermeldung im Wortlaut und, ist nahezu sinnlos und verursacht unnötigerweise Rückfragen, welche für alle langweilig sind. Fehlermeldungen von *Dr. Watson* sind ebenfalls untauglich zur Fehlerbehebung. Beschaffen Sie sich in diesem Falle die Fehlermeldung im Original, also die ursächliche Meldung. Da es diese Witzfigur, Dr. Watson, nach meinem Informationsstand ausschliesslich in **Windows NT** gibt, kontaktieren Sie zuerst Ihren Netzwerkadministrator. Anhänge jeglicher Art sind auch in News-Gruppen verpönt und eigentlich gar nicht erlaubt.

Das **Spotlight-Forum** für MS Word erreichen Sie unter: Online-Treff rund um MS Word

Microsoft unterhält zahlreiche News-Gruppen, auch in deutscher Sprache, welche von vielen fachkompetenten und hilfsbereiten Leute besucht werden. Beiträge für News-Gruppen werden in Outlook Express oder vergleichbaren Anwendungen geschrieben und gelesen. Hier die zwei wichtigsten deutschsprachigen **News-Gruppen** im Zusammenhang mit MS Word. Wenn Sie einer der Links klicken, erfolgt die Einrichtung unter Outlook Express vollautomatisch news://msnews.microsoft.com/microsoft.public.de.word für **allgemeine Fragen** und news://msnews.microsoft.com/microsoft.public.de.word.vba für Fragen, welche Word VBA betreffen. Bitte respektieren Sie diese Unterscheidung.

URL
http://mypage.bluewin.ch/reprobst/WordFAQ/Schluss.htm

René Probst, Mai 2000